Robert Müller

Analyse, Konzeption und Optimierung einer Social Software-Kollaborationsplattform

DIPLOMARBEIT

HOCHSCHULE MITTWEIDA

UNIVERSITY OF APPLIED SCIENCES

Fachbereich Informationstechnik & Elektrotechnik

Mittweida, 2010

Robert Müller

Analyse, Konzeption und Optimierung einer Social Software-Kollaborationsplattform

eingereicht als

DIPLOMARBEIT

an der

HOCHSCHULE MITTWEIDA

UNIVERSITY OF APPLIED SCIENCES

Fachbereich Informationstechnik & Elektrotechnik

Mittweida, 2010

Erstprüfer: Prof. Dr. rer. nat. Sergej Alekseev

Zweitprüfer: Dipl.-Wirtsch.-Inf. Stefanie Städter (FH)

Vorgelegte Arbeit wurde verteidigt am: 27.09.2010

Bibliographische Beschreibung:

Müller, Robert:
Analyse, Konzeption und Optimierung einer Social Software-Kollaborationsplattform.
Dresden, Hochschule Mittweida, Fachbereich Informationstechnik & Elektrotechnik, Diplomarbeit, 2010

Referat:

Ziel der Diplomarbeit ist es, eine Social Software-Kollaborationsplattform zu konzeptionieren, zu analysieren und zu optimieren.
Dafür wird zuerst die theoretische Grundlage geschaffen, indem wichtige Begriffe, wie Web 2.0, Social Software und Enterprise 2.0, erläutert und eingeordnet werden. Anschließend werden Anforderungen an eine Social Software-Kollaborationsplattform aufgestellt, wobei der Fokus auf die Social Software-Funktionalitäten gelegt wird. Darauf aufbauend werden drei ausgewählte Produkte evaluiert und eine Bewertungsmatrix erstellt. Die anschließende Optimierung findet anhand des Microsoft SharePoint Servers 2010 statt.

Vorwort

Die vorliegende Diplomarbeit wurde in der Firma T-Systems Multimedia Solutions GmbH in Dresden angefertigt. Dort fand die Bearbeitung in der Business Unit Microsoft Business Solutions statt.

Für die Ermöglichung dieser Diplomarbeit möchte ich mich bei Prof. Dr. rer. nat. Sergej Alekseev und Dipl.-Wirtsch.-Inf. Stefanie Städter (FH) bedanken. Ebenfalls erwähnt sei René Bardehle, der mir jeder Zeit beratend, vor allem bei Formulierungsfragen, zur Seite stand. Ein Besonderer Dank gilt meiner Familie, die mich während der Bearbeitung dieser Arbeit und im kompletten Studienzeitraum bei jeder Herausforderung unterstützt hat.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Ajax	Asynchronous JavaScript and XML
API	Application Programming Interface
ASP	Active Server Pages
CMS	Content Management-System
CSCW	Computer Supported Cooperative Work
CSS	Cascading Style Sheets
DLL	Dynamic Link Library
DOM	Document Object Model
HTML	Hypertext Markup Language
MOSS	Microsoft Office SharePoint Server
PDF	Portable Document Format
RSS	Really Simple Syndication
URL	Uniform Resource Locator
WCMS	Web Content Management System
WWW	World Wide Web
WYSIWYG	What You See Is What You Get
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language

Glossar

Backend

In der Informationstechnik wird oft eine Schichteneinteilung von Systemen vollzogen, wobei das Backend immer näher an der Verarbeitung oder der Ausgabe ist. (siehe Frontend)

Content Management-System

Ein Content Management-System (kurz CMS) ist ein System, mit dem gemeinschaftlich Inhalt erstellt, bearbeitet und organisiert werden kann.

Folksonomy

Als Folksonomy wird die Klassifikation bezeichnet, die durch Social Tagging entsteht. Es handelt sich also dabei um die entstehende Sammlung der Tags.

Frontend

In der Informationstechnik wird oft eine Schichteneinteilung von Systemen vollzogen, wobei das Frontend immer näher am Benutzer bzw. der Eingabe ist. (siehe Backend)

Makro

Ein Makro ist eine Gruppe von Kommandos, die nach der Aktivierung automatisch hintereinander abgearbeitet werden.

Plug-in

Abgeleitet vom englischen to plug in (dt. „einstöpseln“) stellt ein Plug-in ein Computerprogramm dar, das in ein anderes Softwareprodukt eingeklinkt werden kann, um dessen Funktionsspektrum zu erweitern.

Tablet PC

Sind mobile Computer, die direkt auf dem Bildschirm mit dem Finger oder einem speziellen Stift bedient werden können. Sie können unter anderem als Notizblock oder zum Surfen im Internet verwendet werden.

Tag

Die bei der gemeinschaftlichen Verschlagwortung mit Hilfe von Social Software erzeugten Deskriptoren werden als Tags bezeichnet.

Tag Cloud

Ist eine Methode zur Informationsvisualisierung, bei der eine Liste aus Tags, oft alphabetisch sortiert, flächig angezeigt wird, wobei einzelne unterschiedlich gewichtete Wörter größer oder auf andere Weise hervorgehoben dargestellt werden.

Taxonomie

Die Klassifizierung von Objekten eines gewissen Bereichs nach bestimmten Kriterien wird als Taxonomie bezeichnet. Dabei entstehen Kategorien oder Klassen.

1 Einleitung

1.1 Einordnung und Abgrenzung

Das Internet hat in den letzten 15 Jahren durch technologische Weiterentwicklungen und dem exponentiellen Wachsen von Nutzerzahlen starke Veränderungen durchlaufen. Wurde es Mitte der 90er Jahre noch vorrangig von einem elitären Kreis technisch affiner Menschen im universitären und wissenschaftlichen Umfeld zum statischen Informationsaustausch genutzt, so wird es heute von immer mehr Personen für unterschiedlichste Anwendungsgebiete verwendet. Mittlerweile nutzen zwei Drittel aller Deutschen das Internet und verbringen dabei mehr als 2 Stunden pro Tag im Netz. (BITKOM 2009)

Mit dem Wandel zu einem „Mitmach-Web“ für jedermann wurden neue Ansätze und Technologien entwickelt, die unter dem Schlagwort „Web 2.0“ zusammengefasst werden. Eine zentrale Rolle nehmen dabei Anwendungen ein, die menschliche Kommunikation, Interaktion und Zusammenarbeit über das Internet unterstützen. Diese Anwendungstypen werden Social Software (dt. soziale Software, wird nur sehr selten übersetzt) genannt.

Eines der bekanntesten Beispiele für Social Software ist Wikipedia. Im Januar 2001 als freie Online-Enzyklopädie gestartet, erfreute es sich schnell größter Beliebtheit und gehört inzwischen zu den 10 populärsten Seiten im Internet. (Alexa 2010a) Das Lexikon ist heute in 240 Sprachen verfügbar und umfasst weltweit rund 14 Millionen Artikel. (Wikipedia 2010)

Da sich die Konzepte des „Web 2.0“ als nützlich erwiesen haben und die Mitarbeiter von Unternehmen im Privatgebrauch an die Bedienung der neuen Anwendungen gewöhnt haben, wird vermehrt versucht, Social Software auch im Unternehmenskontext einzusetzen. In diesem Zusammenhang wurde auch der Begriff des „Enterprise 2.0“ geprägt. Insbesondere im Bereich der Kollaboration hat sich herausge-

stellt, dass der Einsatz von Social Software vorteilhaft ist. Aus diesem Grund haben Unternehmen, wie IBM[1] und Microsoft[2], ihre Kollaborationsplattformen mit diesen Anwendungstypen erweitert. In dieser Diplomarbeit werden jene Produkte, die ihren Schwerpunkt auf Kollaboration mit Hilfe von Social Software setzen – im Weiteren Social Software-Kollaborationsplattformen genannt – untersucht.

1.2 Motivation und Zielsetzung

Das Intranet ist eines der wichtigsten Bestandteile der internen Unternehmenskommunikation und -zusammenarbeit. Klassische Intranet-Portale, welche nur als Informations- und Dokumentenverteiler genutzt wurden, werden durch fortschrittliche Kollaborationsplattformen ersetzt oder erweitert. Der Autor dieser Diplomarbeit ist bereits bei Arbeiten für das Studium mit Social Software in Kontakt gekommen. Dabei weckten die anwenderfreundliche Bedienung der Anwendungen und der offensichtliche Mehrwert für kollektive Zusammenarbeit reges Interesse. Der firmeninterne Einsatz solcher Softwaresysteme birgt ein enormes Potential. Die Herausforderung ist dabei, die richtige Software für den benötigten Anwendungszweck zu finden. Plattformen, die mehrere Social Software-Typen zusammenführen und dabei vernetzen, bieten in diesem Fall einen eindeutigen Vorteil.

Motivation für diese Diplomarbeit bildet in erster Linie das Anliegen, ein Thema zu bearbeiten, welches aktuell in der Wirtschaft diskutiert wird und dabei auf moderne Technologien und Konzepte setzt. Social Software im Unternehmenseinsatz erfüllt genau diese Anforderungen.

Das Ziel der Arbeit ist es, zu untersuchen, wie weit fortgeschritten die Integration von Social Software Anwendungen in bekannten Kollaborationsplattformen ist. Dabei sollen die Anwendungen einzeln und im Zusammenspiel miteinander bewertet werden. Da die Bearbeitung der Diplomarbeit in der Business Unit „Microsoft Business Solutions“ stattfindet, und dort der Microsoft SharePoint Office Server

[1] IBM Lotus Connections
[2] Microsoft SharePoint Server 2010

ein zentrales Produkt ist, wird dieser auch bei der Evaluierung im Mittelpunkt stehen. Mit Hilfe einer eigenentwickelten Erweiterung sollen ausgewählte Schwachpunkte für den projektbezogenen Unternehmenseinsatz beseitigt werden.

1.3 Gliederung

Im ersten Teil werden für die Diplomarbeit relevante Grundlagen erklärt. Ferner werden Entwicklungen und Erkenntnisse des interdisziplinären Forschungsgebiets Computer Supported Cooperative Work (oft auch Computer Supported Collaborative Work, im Folgenden mit CSCW abgekürzt) und Groupware betrachtet. Anschließend werden die Konzepte, Anwendungen und Technologien hinter dem Begriff Web 2.0 zusammengefasst. Beides sind Grundlagen für die anschließend folgende Einführung in das Thema Social Software. Abgeschlossen wird dieser einleitende Teil mit einem Überblick in den Einsatz von Social Software in Unternehmen und der Definition von „Social Software-Kollaborationsplattform“.

Im nächsten Abschnitt werden Anforderungen an eine Social Software-Kollaborationsplattform skizziert, hierbei wird der Fokus auf die Social Software gelegt. So findet in zwei Teilen: erst eine anwendungsübergreifende und dann eine anwendungsspezifische Anforderungsbetrachtung statt.

Im Kapitel Evaluierung erfolgt die Bewertung von ausgewählten Produkten. Diese erfolgt auf der Grundlage des zuvor ausgearbeiteten Anforderungskataloges. Die Erkenntnisse der Betrachtung der einzelnen Vertreter werden in eine Bewertungsmatrix eingetragen. Das so resultierende Ergebnis wird abschließend ausgewertet.

Nach der Evaluierung soll mit Hilfe des Ergebnisses der Bewertung eine gezielte Verbesserung am gewählten Vertreter Microsoft SharePoint Server 2010 realisiert und dokumentiert werden.

Zum Schluss werden die Erkenntnisse und Ergebnisse der Arbeit zusammengefasst und mit einem Ausblick in zukünftige Entwicklungen im Bereich der Social Software-Kollaborationsplattformen wird die Diplomarbeit abgerundet.

2 Grundlagen einer Social Software-Kollaborationsplattform

Um die Intention des Begriffs Social Software-Kollaborationsplattform zu vermitteln, bedarf es der Erklärung zentraler Zusammenhänge zwischen: „Web 2.0“, „Social Software“, „Enterprise 2.0“ und „CSCW“ bzw. „Groupware“.

2.1 CSCW und Groupware

Bereits in den achtziger Jahren wurde interdisziplinär in Grenzbereichen der Informatik, Betriebswirtschaftslehre, Psychologie und Soziologie an Software zur Unterstützung von Kooperation in Teams und Organisationen geforscht. Dies erfolgte unter den Schlagworten Computer Supported Cooperative Work (CSCW) oder Groupware. (vgl. Koch und Richter 2009, S. 16)

„Unter Rechnergestützter Gruppenarbeit oder Computer-Supported Cooperative Work (CSCW) versteht man einen multidisziplinären Forschungsbereich, der sich mit dem Verstehen sozialer Interaktion sowie der Gestaltung, Implementation [sic] und Evaluierung von technischen Systemen zur Unterstützung sozialer Interaktion beschäftigt.“ (Gross und Koch 2007, S. 10)

Im Forschungsgebiet CSCW wurden also die Möglichkeiten, wie die Zusammenarbeit von Menschen durch Kommunikations- und Informationstechnologie unterstützt werden können, untersucht. Anwendungen, die in diesem Sinne entwickelt worden, werden unter dem Begriff Groupware zusammengefasst.

In der Unternehmenspraxis wird Groupware oft mit Microsoft Outlook/Exchange bzw. in diese Klasse gehörigen Client/Server-Produkten, die auf dem MAPI-Protokoll basieren, gleichgesetzt. Dies umfasst E-Mail, (gemeinsame) Terminkalender, (gemeinsame) Adressbücher und (gemeinsame) ToDo-Listen. Das sind aber nur Beispiele für asynchrone Zusammenarbeit, zu denen auch Werkzeuge zum Verwalten gemeinsamer Datenbestände (Informationsräume) mit Awareness[3]-

[3] dt. Bewusstheit, wird in der Fachliteratur nicht übersetzt

Funktionalität gehören. Beispiele für synchrone Zusammenarbeit sind Instant-Messaging-Anwendungen, Konferenzsysteme und (synchrone) Gruppeneditoren.

Neben dem Einordnen von Groupware nach der Synchronität sei noch die Klassifikation nach der primär unterstützten sozialen Interaktion nach Teufel genannt, siehe dazu Abbildung 2-1.

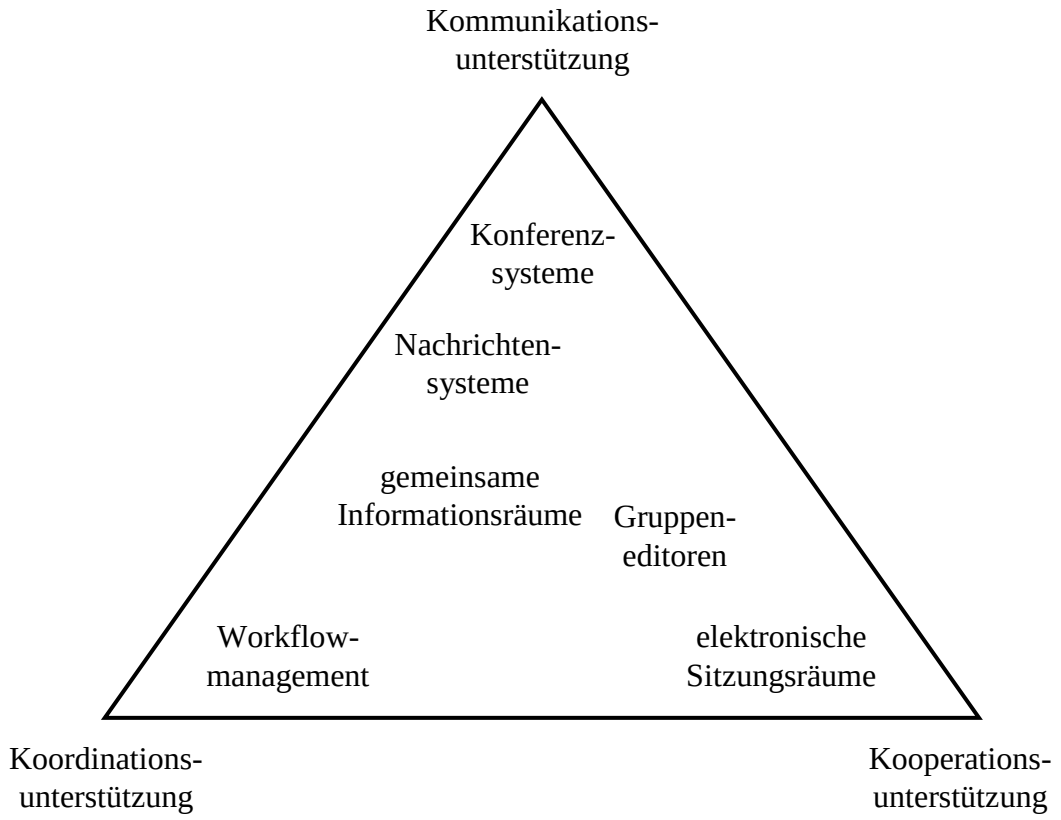


Abbildung 2-1: Klassifikation von Groupware nach Interaktionstypen (Teufel, 1995)

Ausgewählte Erkenntnisse der CSCW-Forschung mit Arbeit von Groupware sind auch auf den Einsatz von Social Software anwendbar. Ein zentraler Aspekt von Groupware ist Awareness, die Grundlage der effizienten und erfolgreichen sozialen Interaktion, d.h. der gegenseitige Austausch von Informationen über Aktivitäten und Status. Gutwin, Greenberg und Roseman (1996) differenzieren vier elementare und notwendige Arten von Awareness-Informationen, die Gross und Koch wie folgt zusammenfassen:

- *„**Informelle Awareness:** Informationen über die Präsenz, Aktionen und Absichten anderer Benutzerinnen und Benutzer sowie die Erreichbarkeit anderer Personen im realen und im elektronischen Raum.*

- ***Soziale Awareness:** Informationen über die Interessen, Aufmerksamkeit und den emotionalen Zustand anderer Benutzerinnen und Benutzer, welche in persönlichen Gesprächen typischerweise über Augenkontakte, Gesichtsausdruck und Körpersprache wahrgenommen werden.*

- ***Awareness über Gruppenstruktur:** Informationen über die Gruppe sowie ihre Mitglieder und deren Rollen, Verantwortlichkeiten, Stati [sic] und Positionen.*

- ***Awareness über den Arbeitsbereich (workspace awareness):** Informationen über die Interaktion der anderen Benutzerinnen und Benutzer mit dem gemeinsamen Arbeitsbereich und den enthaltenen Artefakten wie beispielweise über die Anwesenheit und Identität von anderen im gemeinsamen Arbeitsbereich, deren Aktionen, deren Absichten, usw.“* (Gross und Koch 2007, S. 25)

Wie in Abbildung 2-2 zu sehen ist, überschneiden sich die vier Typen von Awareness bei der Gruppenarbeit. Es findet Informationsaustausch und Interaktionen über die Grenzen hinweg statt.

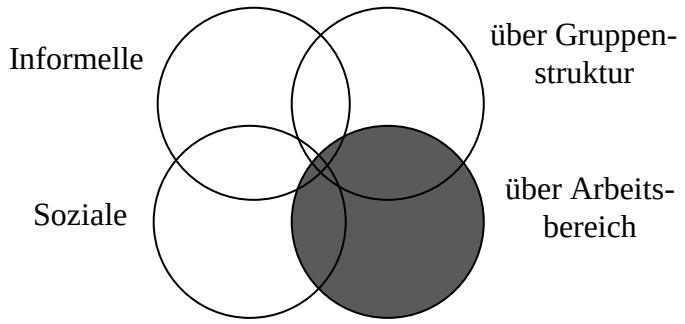


Abbildung 2-2: Arten von Awareness nach Gutwin, Greenberg und Roseman (1996)

Somit ist der charakteristische Unterschied zwischen Groupware und anderer Software zur Unterstützung mehrerer Benutzer, wie z.B. Mehrbenutzerdatenbanksyste-

me, der explizite Versuch mit Hilfe von Awareness-Informationen die Isolation der Benutzer voneinander zu beseitigen. Statt die Benutzer voreinander zu schützen und zu verstecken, wird beim Benutzer ausdrücklich eine Bewusstheit dafür generiert, dass er einer Gruppe zugehörig ist.

Als zweiter wichtiger Erfolgsfaktor von Groupware wird deren Anpassbarkeit gesehen. Der Erfolg von Groupware ist nur möglich, wenn eine kritische Masse von Benutzern aktiviert werden kann. Da sich aber die Anforderungen jede Gruppe und jedes individuelle Gruppenmitglied unterscheiden können, sind spezifische Lösungen unabdingbar. Nur wenn Groupware sehr generisch und/oder anpassbar ist, können Benutzer und Gruppen diese auf unterschiedliche Art und Weise nutzen. Dabei sollte immer angestrebt werden, dass die Endnutzer die Software selbst erweitern und an ihre Bedürfnisse anpassen können. End-User-Development (dt. Endbenutzer-Entwicklung) ist ein Teilgebiet der Informatik, das sich genau mit dieser Thematik beschäftigt.

Abschließend sei noch die Erkenntnis der CSCW-Forschung genannt, dass eine zyklische, kontinuierliche und evolutionäre Entwicklung für den Erfolg eines Groupware-Systems maßgebend ist. (vgl. Koch und Richter 2009, S. 19)

2.2 Das Web 2.0

Eine konkrete Definition für den Begriff Web 2.0 gibt es nicht, dennoch ist der Term im Internet weit verbreitet.[4] Für eine Erklärung des Ausdrucks ist ein Exkurs in dessen Entstehungsgeschichte notwendig. Mitte der 90er Jahre stiegen die Benutzerzahlen des Internets explosionsartig an und mit zunehmender Popularität wuchs auch das wirtschaftliche Interesse. Es entstanden neue Unternehmen, die ausschließlich im Internet agierten und dort ihre Waren und Dienstleistungen anboten. Über einen Börsengang beschafften sich viele dieser Unternehmen zusätzliches Kapital. Die neuen technologischen Entwicklungen entfachten eine immer größere werdende Nachfrage nach entsprechenden Aktien. Infolge dessen kam es zu einem

[4] Vgl. ungefähr 448.000.000 Suchergebnisse bei http://www.google.de, Ergebnis am 05.03.2010

Boom, der aufgrund der Internet-Domain-Endung „.com“ (engl. für Commercial) auch Dotcom-Boom genannt wurde. Im Jahr 2000 stoppte dieser Aufschwung abrupt mit einem Börsencrash. Viele der zuvor gegründeten Internetunternehmen gingen insolvent oder mussten um ihre Existenz kämpfen. In diesem Zusammenhang wird rückblickend vom Platzen der Dotcom-Blase gesprochen.

Der Begriff Web 2.0 erlangte erstmals im Jahr 2004 im Rahmen der Planungen zu einer Konferenz des Verlages O’Reilly Media und des Konferenzveranstalters MediaLive an Bedeutung. Intention dieser Veranstaltung war es, den Stand der Entwicklungen des World Wide Webs (kurz: Web oder WWW) zu erfassen. Zu diesem Zeitpunkt sind Internet-Unternehmen, wie Google, Amazon oder eBay, gestärkt aus der Krise hervorgegangen und die Organisatoren der Konferenz suchten nach Unterscheidungsmerkmalen zu den in Konkurs gegangenen Firmen. Nach Ende der Tagung herrschte Uneinigkeit über eine Definition des Begriffs und Web 2.0 lief Gefahr, ein Modewort der Marketingabteilungen zu werden. Mit dieser Erkenntnis verfasste Tim O’Reilly den Artikel „What is Web 2.0?“ (vgl. O’Reilly 2005), welcher heute als die einflussreichste Beschreibung gilt. Aus heutiger Sicht können darauf aufbauend unter anderen[5] folgende Konzepte und Eigenschaften (vgl. Alby 2008 und Friedman 2008) zusammengefasst werden:

Das Web als Service-Plattform: Die Web 2.0-Firmen nutzen das Internet als Plattform, auf der sie ihre Dienste anbieten. D.h. ihre Anwendungen sind komplett webbasierend und müssen nicht mehr auf einem Rechner installiert und gepflegt werden. Mit der Hilfe von oft einfachen Programmierschnittstellen, so genannte APIs (engl. für Application Programming Interfaces), kann auf die Dienste geräte- und betriebssystemunabhängig zugegriffen werden. So werden klassische Desktopanwendungen, wie Terminplanung und Text- und Bildverarbeitung im Web angeboten.

Frei Kombinierbare Datenquellen und Datentransformationen: Wichtigstes Gut der Seiten sind ihre aggregierten Daten. Die Inhalte sind zum Teil teuer gekauft oder auch von den Anwendern erstellt. Die Webanwendungen stellen für einen

[5] Wirtschaftliche (Social Commerce) Konzepte wie „Log Tail“ werden nicht betrachtet

leichten Zugang der Daten offene APIs zur Verfügung. Viele Anwendungen nutzen dies, um Daten mit anderen Quellen zu kombinieren.

Weisheit der Massen und Architektur der Partizipation: Mit dem Ziel das Wissen aller Nutzer einzubeziehen, wird Social Software eingesetzt. Die neuen Webanwendungen[6] sind so konzipiert, dass die Benutzer aktiv werden und selbst Inhalte gestalten. Dabei entsteht ein sogenannter Netzwerkeffekt, denn umso mehr Benutzer die Anwendungen aktiv verwenden, desto informativer und produktiver werden diese für alle. Bei den von den Nutzern erstellten Webinhalten wird vom User Generated Content (dt. nutzergenerierten Inhalt) gesprochen.

Perpetual Beta: Die meisten Applikationen sind in einem dauerhaften Beta-Status. Dabei werden die Nutzer bewusst oder unbewusst in die Entwicklung einbezogen. Neue Versionen der Software werden möglichst früh und oft veröffentlicht. Das Echtzeitverhalten der Anwender und der Umgang mit den neuen Funktionen werden fortlaufend analysiert und in die Weiterentwicklung einbezogen. Somit haben die Benutzer mit ihrer Art der Nutzung der Software direkten Einfluss auf deren Entwicklung.

Einfachheit und Minimalismus: Hinter den Anwendungen stehen oft überschaubare, schlichte oder gar minimalistische Ansätze. Ein weit verbreitetes Paradigma ist „Focus on Simplicity“ (dt. Fokus auf Schlichtheit). Anwendungen werden auf die wichtigsten Funktionen beschränkt, damit die Anwender Webdienste schnell und effizient bedienen können, ohne sich dabei in komplexe Bedienungen einarbeiten zu müssen. Außerdem werden Lightweight Programming Models (dt. leichtgewichtige Programmieransätze) verwendet. So ermöglichen „leichte“ APIs und Technologien, wie RSS (Really Simple Syndication), lockere Verbindungen zwischen zwei Diensten.

Software Above the Level of a Single Device: Die Webanwendungen werden mit dem Anspruch entwickelt, auf allen mit dem Internet vernetzten Geräten zu funkti-

[6] Vgl. Auflistung und Erklärung der Web 2.0 Social Software-Anwendungen im Abschnitt 2.3.2, S. 14

onieren. In den letzten Jahren rückt die Kompatibilität mit mobilen Geräten, wie Smartphones und Tablet PCs[7] immer mehr in den Fokus.

Rich User Experiences: Früher konnten Webseiten immer nur im Ganzen aktualisiert werden, was eine umständlichen Benutzerführung zur Konsequenz hatte. Mit Hilfe von neuen Technologien, wie z.B. Ajax (Asynchronous JavaScript and XML), das das partielle Aktualisieren von Webseiten ermöglicht, wird eine ähnliche Benutzungserfahrung wie bei Desktopanwendungen geboten.

Anhand dieser Auflistung ist erkennbar, dass der Begriff Web 2.0 zwar eine Anlehnung an die Versionsnummerierung in der Software-Entwicklung ist, aber nicht nur technische Meilensteine implizieren soll. Vor allem die Entwicklungen im Bereich der Benutzer sind herauszuheben. Der individuelle Nutzer steht jetzt im Mittelpunkt des Geschehens und wird an verschiedensten Stellen zur Partizipation ermutigt. Die Grenzen zwischen Autoren und Lesern verschmelzen zunehmend. Die Anwender erstellen aber nicht nur Inhalte, wie Weblogs und Rezessionen, sondern auch Kontext, wie zum Beispiel beim Bewerten und Kategorisieren von Videos.

Eine Erklärung dafür, dass sich die Nutzer im Internet beteiligen, ist, dass nicht versucht wird die Benutzer in Gruppen und Communities einzuteilen. Stattdessen steht der individuelle Nutzer immer im Mittelpunkt des Geschehens. Die Benutzer verbinden sich untereinander, wodurch eine neue Art der Kommunikation und Vernetzung entsteht. In den Netzen können sich jedoch Cluster bilden, die Webnutzer mit Gemeinsamkeiten zusammenführen. Diese gelten aber als sehr offen. Innerhalb der Netzwerke können die Anwender individuell und getrennt voneinander agieren und je nach Bedarf bereits vorhandene Kontakte nutzen oder neue Kontakte zu bisher unbekannten Personen ausbilden. So wird der Austausch und der fließende Übergang zwischen verschiedenen Clustern ermöglicht. Der Fokus auf die persönlichen Bedürfnisse, wie Kommunikation, Selbstdarstellung, Dokumentation und Kategorisierung (vgl. Koch und Richter 2009, S. 3) und der Verzicht der Unterordnung unter Gruppeninteressen senkt die Mitmachhürde und trägt so zu einer höheren Beteiligung bei.

[7] Vgl. Erklärung siehe Glossar

Usability ist ein weiterer Schlüsselfaktor für die Motivation der Nutzer. Dank einfachem Aufbau der Anwendungen und meist intuitiven Bedienmöglichkeiten ist die Motivation zum Experimentieren recht hoch. Zum Beispiel kann ein Blog[8] mit WordPress[9] innerhalb weniger Minuten eingerichtet und ein erster Artikel erstellt werden. Zusätzlich ist ein Großteil dieser Anwendungen als freie Open Source-Software verfügbar. All das führt dazu, dass immer mehr Laien selbst Blogs erstellen und verwalten. Durch Umstellung auf die individuellen Bedürfnisse und einem vermehrten Partizipieren von Nutzern wird auch von einem „Mitmach-Web" gesprochen.

2.3 Social Software

Erste Erwähnungen von Social Software können bis in die neunziger Jahre zurückverfolgt werden (vgl. Allen 2004), dessen ungeachtet hat sich der Begriff erst in den letzten Jahren etabliert. Eine einfache Umschreibung für Social Software lieferte Weblog-Pionier Tom Coates 2005:

„Social Software can be loosely defined as software which supports, extends, or derives added value from, human social behavior." (Coates 2005)

Dies ist eine verhältnismäßig lockere Definition, da dieser Ansatz quasi auf alle Software-Typen passt, die bei zwischenmenschlichen Interaktionen zum Einsatz kommen können. Das Einsatzszenario ist nicht wirklich neu. Wie im Kapitel CSCW und Groupware bereits aufgezeigt, wird in diesem Bereich bereits seit Jahrzehnten geforscht. Wirkliche Popularität erreichte der Begriff Social Software jedoch erst mit dem Aufstieg verschiedener Webanwendungen, die aus heutiger Sicht dem Web 2.0 zugeordnet werden können. So sind die Ursprünge von Social Software vor allem in den sozialen und technischen Entwicklungen des Web 2.0 zu suchen. Um diese Relation zu verdeutlichen, wird Social Software im Zuge dieser Arbeit definiert als:

[8] Eine ausführliche Beschreibung von Blog erfolgt in Abschnitt 3.2.1.
[9] WordPress ist eine freie Software zum erstellen von Blogs. Vgl. http://www.wordpress.org/, verfügbar am 28.01.2010

„Anwendungssysteme, die unter Ausnutzung von Netzwerk- und Skaleneffekten, indirekte und direkte zwischenmenschliche Interaktion (Koexistenz, Kommunikation, Koordination, Kooperation) auf breiter Basis ermöglichen und die Identitäten und Beziehungen ihrer Nutzer im Internet abbilden und unterstützen." (Koch und Richter 2009, S. 12)

Das Internet ist hierbei nicht als scharfe Deklaration zu verstehen, vielmehr sind damit im weiteren Sinne internetbasierte Netzwerke gemeint. Betrachtet man Social Software als Teilanwendungen des Web 2.0, so sind auch viele Konzepte und Erkenntnisse auf Social Software übertragbar.

2.3.1 Unterschiede und Gemeinsamkeiten zu Groupware

Sucht man einen Unterschied zwischen Social Software und Groupware, kann man diesen bei den verschiedenen Anwendergruppen finden. So spricht Social Software explizit Communities bzw. Netzwerke an. Ein zentrales Konzept des Web 2.0 war es, dass die Nutzer sich selber untereinander vernetzen und so lose Netzwerke bilden. Social Software hilft bei der Bildung solcher Netzwerke oder unterstützt sie. Im Mittelpunkt steht der einzelne Benutzer. Im Gegensatz dazu besteht die Anwendergruppe bei Groupware aus Teams bzw. Gruppen. So liegt bei Groupware der Fokus, wie man aus dem englischen Wort „work" im Akronym CSCW herleiten kann, auf Arbeit. Dementsprechend haben sich klassische Groupware-Anwendungen ausschließlich mit Hauptaugenmerk auf Arbeit in Gruppen und auch unter klaren Vorgaben weiterentwickelt. Hingegen zeichnen sich Communities nicht unbedingt durch einen strickten Fokus auf Arbeit aus. Ganz im Gegenteil – das oft kreative Umfeld nutzt Social Software meist frei von Vorgaben und experimentell. Dadurch bedingt ist unter dem Begriff Social Software ein breites Spektrum eigenständiger Anwendungen mit neuem Funktionsumfang entstanden.

Neben diesen Unterschieden bestehen aber auch Gemeinsamkeiten. Ein Beispiel dafür ist die kontinuierliche Weiterentwicklung, was ein Erfolgskriterium für Groupware ist. Diese wird unter dem Schlagwort Perpetual Beta für Social Soft-

ware praktiziert. Aber auch die anderen Erfolgsfaktoren von Groupware, wie z.B. gezieltes Aufheben von Isolation der Nutzer und Anpassbarkeit, sind genauso auf Social Software übertragbar. Zudem beschäftigt sich die CSCW-Forschung zunehmend mit der Unterstützung von lose gekoppelten Gruppen. Wird Social Software nun in Unternehmen eingesetzt, können diese als Teilgebiet der Groupware betrachtet werden.

2.3.2 Charaktersierung von Social Software

Die Einteilung von Social Software nach deren Charakteristika ermöglicht einen tieferen Einblick in die Chancen, die sich durch deren Einsatz bieten. Grundlage dafür stellt zunächst eine Aufschlüsselung der bekannten Anwendungstypen. Zu den in dieser Arbeit betrachteten Anwendungsklassen gehören:

- **Wikis**, Hypertextsysteme für Webseiten, die nicht nur gelesen, sondern auch gemeinsam von Nutzern versionsgesichert erstellt und modifiziert werden können,
- **Weblogs**, meist öffentlich einsehbare und kommentierbare webbasierte Tagebücher,
- **Microblogs**, Sonderform von Blogs, über die nur kurze Nachrichten verschickt werden können,
- **Social Network Services**, Dienste zur Bildung und Pflege von sozialen Netzwerken,
- **Social Bookmarking**, Anwendungen zum Erstellen und Austauschen von Weblesezeichen,
- **Social Tagging**, Dienste für eine gemeinsame Verschlagwortung von Inhalten,
- **Instant Messaging**, Anwendungen zur textbasierter Echtzeitkommunikation zwischen Nutzern.

Auf die verschiedenen Anwendungsklassen wird im Kapitel Anforderungen noch explizit eingegangen.

Es gibt aber noch weitere Möglichkeiten zur Strukturierung von Social Software. Eine geeignete Strukturierung liefern zum Beispiel Koch und Richter. Sie betrachten den Einsatzzweck von Social Software und unterscheiden dabei folgende drei Basisfunktionen (vgl. Koch und Richter 2009, S. 12):

- **Informationsmanagement:** Hierzu gehören Möglichkeiten für das Finden, Bewerten und Verwalten von (online verfügbaren) Informationen.
- **Identitäts- und Netzwerkmanagement:** Umfasst die Darstellung von Aspekten der eignen Identität im Internet sowie das Knüpfen und Pflegen von Kontakten.
- **Kommunikation:** Beinhaltet Fähigkeiten für die direkte und indirekte Kommunikation zwischen den Benutzern.

Dabei beschränken sich die meisten Social Software-Anwendungen nicht nur auf eine Basisfunktion. Vielmehr bieten einige dieser Anwendungen Teilfunktionen für alle drei Nutzungsarten an.

In Anlehnung an die Einteilung von Groupware nach Interaktionstypen (Abb. 2-1) wurden in Abbildung 2-3 die zuvor gelisteten Social Software Anwendungstypen nach ihren Basisfunktionen eingeordnet. Dies ist eine veränderte Version des „Social Software“-Dreieck von Richter und Koch (2009, S. 14).

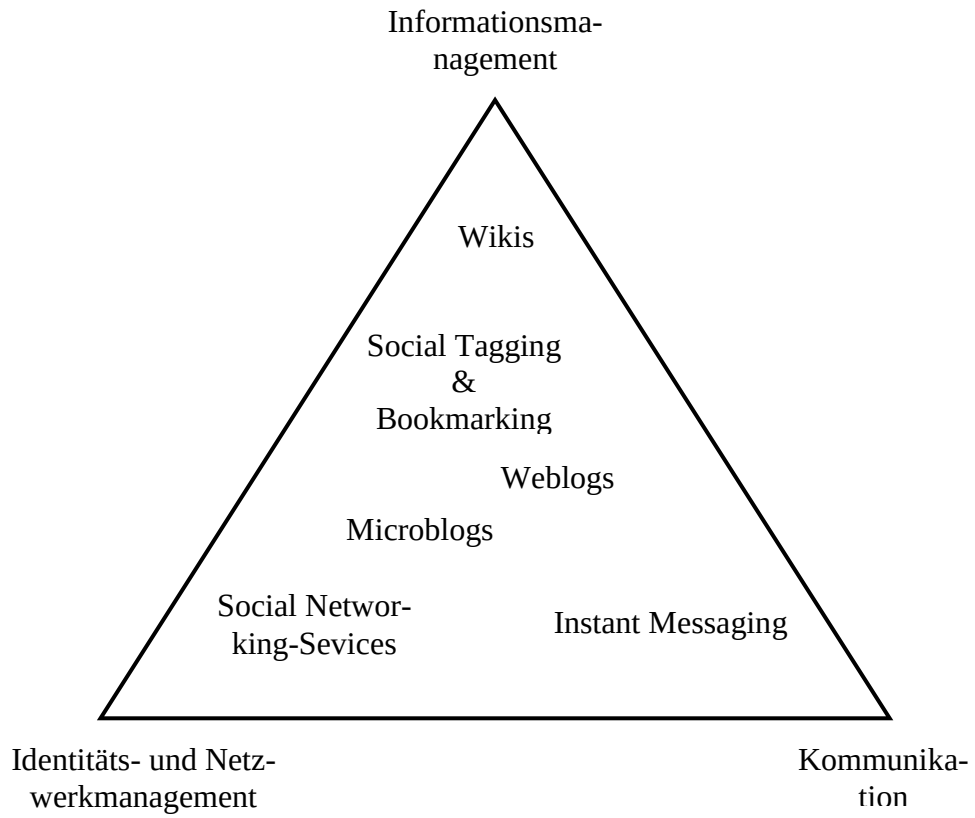


Abbildung 2-3: Das „Social Software-Dreieck“

Zusätzlich sei die Strukturierung von McAfee (2006a) genannt. Unter dem Akronym SLATES (search, links, authoring, tags, extensibility, signals) fasst er weitere Herausstellungsmerkmale von Social Software zusammen. Richter und Koch haben diese Liste angepasst und wie folgt zusammengefasst:

- *„So einfach wie möglich selbst Beiträge veröffentlichen oder Inhalte editieren können („**authoring**")*
- *Durch Tagging einfach strukturierende Metadaten beitragen können („**tags**")*
- *Durch Annotations- und Verlinkungsmöglichkeiten einfach zusätzliche Inhalte und Metadaten bereitstellen können („**authoring**", „**link**s")*
- *Durch Abonnierungsmöglichkeiten einfach auf neue Inhalte aufmerksam gemacht werden können („**signals**")*

- *Beigetragene Inhalte einfach auffindbar machen („**search**", „**tags**")*
- *Modularer, dienstorientierter und datenzentrierter Aufbau der Anwendungen („**extensions**")"* (2009, S. 14)

2.4 Enterprise 2.0 - Social Software im Unternehmen

Unter dem Schlagwort Enterprise 2.0 (manchmal auch „enterprise social software") hat sich mittlerweile eine eigene Bewegung herausgebildet, die sich mit der Einführung und der Verwendung von Social Software in Unternehmen beschäftigt. Prägend für diesen Begriff war Harvard Business School Professor McAfee, der im Jahr 2006 einen Artikel mit dem Titel „Enterprise 2.0: The Dawnn of Emergent Collaboration" (McAfee, 2006a) veröffentlichte. Später definierte er Enterprise 2.0 folgendermaßen:

„*Enterprise 2.0 is the use of emergent social software platforms within companies, or between companies and their partners or customers."* (McAfee 2006b)

Durch den Einsatz von Social Software-Anwendungen ergeben sich für die Unternehmen völlig neue Chancen. Anwendungen, wie Wikis, tragen dazu bei, dass Wissen im Unternehmen besser gewonnen und erhalten werden kann. Über die Profilverwaltung und Vernetzung der Mitarbeiter können zusätzlich Wissensträger im Unternehmen schneller gefunden werden. So eröffnen Social Software-Anwendungen Möglichkeiten für die Überwindung von Informationssilos. Zusätzlich zur Informationsverteilung kann Social Software neue und verbesserte Möglichkeiten für die Zusammenarbeit schaffen.

Exemplarisch kann mit einer Microblog-Anwendung ein Kommunikationskanal generiert werden, der dem Unternehmen und den Mitarbeitern eine stetige Resonanz für ihre Arbeit liefert. Aus diesem Grund hat Gartner Inc.[10] ein Jahr nach

[10]Vgl. http://www.gartner.com – Die Gartner Inc. gehört zu den weltweit führenden Anbietern für Marktforschung und Analyse im Informationstechnik-Sektor.

McAfees Publikation auch Social Software zu den zehn strategisch wichtigsten Technologien für Unternehmen für die unmittelbaren Jahre ab 2008 erklärt:

„Social Software. Through 2010, the enterprise Web 2.0 product environment will experience considerable flux with continued product innovation and new entrants, including start-ups, large vendors and traditional collaboration vendors. Expect significant consolidation as competitors strive to deliver robust Web 2.0 offerings to the enterprise. Nevertheless social software technologies will increasingly be brought into the enterprise to augment traditional collaboration." (Pettey 2007)

Aus der zunehmend wachsenden Zahl von Studien zum firmeninternen Einsatz von Social Software, wie Richter und Koch (2009) oder Back, Gronau und Klaus (2009), lassen sich wertvolle Erkenntnisse entnehmen. So ist der Erfolg im Unternehmen direkt davon abhängig, wie die Mitarbeiter die Software annehmen. Die Voraussetzungen dafür unterscheiden sich kaum von denen im Internet. Wichtig sind auch hier die Grundprinzipien, wie Usability und Anpassbarkeit der Anwendungen.

Ferner ist der Schlüssel für die effektive Arbeit mit Social Software, dass der richtige Anwendungstyp für das benötigte Szenario verwendet wird. Die anwendungsspezifischen Erfahrungen und skizzierten Anforderungen an Social Software im Bereich Enterprise 2.0 fließen direkt in den weiteren Verlauf dieser Arbeit ein.

2.5 Social Software-Kollaborationsplattform

Bei dem Ausdruck Social Software-Kollaborationsplattform handelt es sich um einen eigens für diese Arbeit gewählten Begriff. Die Intention für die Wahl einer neuen Begriffsdefinition resultiert aus der Tatsache, dass zum einem in den letzten Jahren immer mehr Anwendungssysteme mit einen bestimmten Einsatzszenario entstanden sind, zum anderen es aber keine zufriedenstellende Bezeichnung für diese Anwendungsgruppe gibt. Hintergrund dieser Entwicklung ist, dass die meis-

ten dieser Software-Produkte ursprünglich unterschiedliche Ansätze verfolgt haben und somit verschiedene Grundfunktionalitäten und Bezeichnungen mit sich bringen.

Gemein haben die Anwendungssysteme jedoch, dass sie als Kollaborationsplattform dienen und zu diesem Zweck verschiedene Social Software-Anwendungen und Dienste einsetzen. Als Grundlage dafür fungiert eine Weboberfläche, die als Schnittstelle zwischen den Anwendungen agiert. Damit wird klar, dass sich diese Kollaborationsplattformen von Anwendungs-Suiten unterscheiden, bei denen in der Regel die einzelnen Anwendungen nicht in einer einheitlichen Plattform integriert werden. Eine weitere Gemeinsamkeit der Produkte wird durch das Wort Kollaboration verdeutlicht; alle Social Software-Kollaborationsplattformen sind primär für den Einsatz in produktiven Umgebungen, wie Organisationen bzw. Unternehmen, konzipiert.

Nachstehend ist eine Liste von Produkten aufgeführt, deren aktuelle Versionen stellvertretend als Social Software-Kollaborationsplattformen gesehen werden können.

Social Software-Kollaborationsplattform	Kurzbeschreibung
Microsoft SharePoint Server 2010	Der SharePoint Server ist Microsofts umfangreiche Portal-Kollaborationsplattform, bei der das Unternehmen den Ansatz verfolgt, so viele Funktionalitäten wie möglich zu integrieren (Document Management-System, Enterprise Content Management-System, etc.). War der der SharePoint Server in früheren Versionen noch ohne Social Software-Einbindung, so ist diese in der aktuellen Version ein zentrales Thema.
IBM Lotus Connections	Lotus Connections ist ein Social Software-Paket von IBM. Ab der Version 2.5 wurde unter der Einführung von Community-Bereichen eine verstärkte Verzahnung der einzelnen Anwendungen erreicht. Mit dem gemeinsam Einsatz anderer IBM-Produkte wie Lotus Quickr (ermöglicht Verwaltung von Teambereichen) und Lotus Sametime (für Instant Messaging und andere Kommunikationsanwendungen) kann IBM Lotus Connections zu einer der umfangsreichsten Social Software-Kollaborationsplattform erweitert werden.
Atlassian Confluence	Confluence ist ursprünglich als reine Enterprise-Wiki-Software gestartet und gehört mittlerweile in diesem Bereich zu den Marktführern. Mit den letzten Releases wurde die Social Software-Integration – gerade im Bereich Social Network Services – stark verbessert.
Jive Social Business Software	Die Social Business Software von Jive ist aus Clearspace – einer Forum-Software – entsprungen und hat neben Kollaboration sein Hauptaugenmerk auf Diskussionen., trotzdem wird in der aktuellen Version ein breites Spektrum an Social Software-Diensten angeboten.
Socialtext	Socialtext – von der gleichnamigen Firma – war ehemalig eine erfolgreiche, kommerzielle Wiki-Software, welche in den letzten Jahren mit weiteren Social Software-Anwendungen zu einer Kollaborationsplattform ausgebaut wurde.
Liferay Social Office	Social Office basiert auf der Open Source-Portalsoftware Liferay Portal und bietet ein vorgefertigtes Paket von Social Software-Funktionalitäten.

Tabelle 2-1: Social Software-Kollaborationsplattformen

3 Anforderungen an eine Social Software-Kollaborationsplattform

Für die Bewertung von Social Software-Kollaborationsplattformen ist es unabdingbar zuvor mögliche Anforderungen zu skizzieren. Für die Bearbeitung dieser Arbeit wurde kein spezielles Einsatzszenario festgelegt. Jedoch sind vielfältige Nutzungsszenarien denkbar, mit denen sich die Voraussetzungen an ein betrachtetes Produkt durchaus unterscheiden. Vor diesem Hintergrund werden nur Vorrausetzungen untersucht, die möglichst allgemeingültig sind. Im Fokus der Betrachtung sollen primär die Social Software-Anwendungen stehen, unter diesem Aspekt wird im Folgenden zwischen anwendungsübergreifenden und anwendungsspezifischen Anforderungen unterschieden.

3.1 Anwendungsübergreifende Anforderungen

Zu den anwendungsübergreifenden Anforderungen gehören zum einen diejenigen, die für jede Anwendung vorausgesetzt werden können, zum anderen die, welche das erfolgreiche Zusammenspiel zwischen den Social Software-Komponenten über deren Grenzen hinweg garantieren. Zuerst werden in diesem Sinn die wichtigsten technologischen Voraussetzungen betrachtet, danach folgt eine Auflistung allgemeiner weiterer Anforderungen.

3.1.1 Technologie

Social Software-Kollaborationsplattformen sind webbasierte Software Systeme und die integrierten Social Software-Anwendungen mit dem Web 2.0 stark verzahnt. Bei der Suche nach technologischen Anforderungen stößt man somit unausweichlich auf die Techniken, die die Basis für den Erfolg der Web 2.0-Firmen bildeten. Die entsprechenden Konzepte wurden bereits Mitte bzw. Ende der neunziger Jahre entwickelt. Trotzdem wird in diesem Zusammenhang von neuen Technologien ge-

sprochen, was in erster Linie daran liegt, dass diese in derartiger Form und Ausmaß bis dahin nicht vorhanden waren.

Eine komplette Auflistung aller denkbaren technologischen Anforderungen schadet der Übersichtlichkeit, weswegen bewusst diverse Voraussetzungen weggelassen werden. Dazu gehören beispielsweise Mashups[11], die zwar eine große Bedeutung im Web 2.0 erlangen konnten, aber in Social Software-Kollaborationsplattformen wenig Verbreitung finden. Zusätzlich wird auf technologische Aspekte, wie zum Beispiel die Skalierbarkeit, die zu sehr vom speziellen Einsatzszenario abhängt, verzichtet.

3.1.1.1 Interaktive Web-Anwendungen - Ajax

Das Akronym Ajax (Asynchronous JavaScript and XML) steht nicht für eine neue Technologie, vielmehr verbirgt sich dahinter ein neues Konzept für Webanwendungen, die sich durch ihre interaktive Bedienung auszeichnen. Anders als der Name vorgibt, sind diese Anwendungen nicht unbedingt asynchron und auch der Einsatz von JavaScript und XML ist nicht zwingend notwendig. (vgl. Ullman, 2007) Ein Mix von clientseitigen Technologien ermöglicht Webanwendungen die Kommunikation mit einem Server, ohne dass dabei die Webseite komplett neu geladen werden muss.

Zunächst wird, wie bei klassischen Anwendungen im Internet üblich, (X)HTML und CSS (Cascading Style Sheets) für eine standardkonforme Seitenstruktur und Darstellung im Browser eingesetzt. Darüber hinaus setzen Ajax-Anwendungen auf das Document Object Model (DOM). Der Zugriff auf diese Spezifikation erlaubt es, dass Inhalt, Struktur und Layout der angezeigten Webseite dynamisch verändert werden können. Allerdings ist das Herzstück einer Ajax-Engine die verwendete Skriptsprache. Diese überwacht die Interaktionen der Benutzer, koordiniert bei Bedarf die Kommunikation zwischen Client und Server, interpretiert die Antworten

[11] Mashups sind Kombinationen aus bestehenden Inhalten oder Daten bzw. Services aus unterschiedlichen Quellen in einer kombinierten und damit neuen Anwendung.

des Webservers und steuert die Manipulation der Webseite. (siehe Abbildung 3-1) Allein schon durch die Kompatibilität mit den wichtigsten Internetbrowsern wird bei der Entwicklung von Webanwendungen meistens auf JavaScript zurückgegriffen. Damit die clientseitige Skriptsprache direkt mit einem Webserver kommunizieren kann, wird das sogenannte XMLHttpRequest-Objekt verwendet. Dieses Objekt bietet zusätzlich einen Modus zur asynchronen Datenübertragung an.

Der fundamentale Unterschied zu klassischen Webanwendungen ist somit, dass Daten im Hintergrund sukzessiv nachgeladen werden können und dabei alle Client-Server-Prozesse für den Nutzer verdeckt bleiben. So können Ajax-Anwendungen genauso zeitnah auf Benutzereingaben reagieren wie Desktop-Anwendungen und vermitteln dadurch eine bei Webanwendungen bisher nicht dagewesene Interaktivität und Benutzbarkeit. Aus diesen Gründen sollte Ajax überall, wo Interaktionen zwischen den Nutzer und den Social Software-Anwendungen stattfinden und es aus technologische Sich Sinn macht, eingesetzt werden.

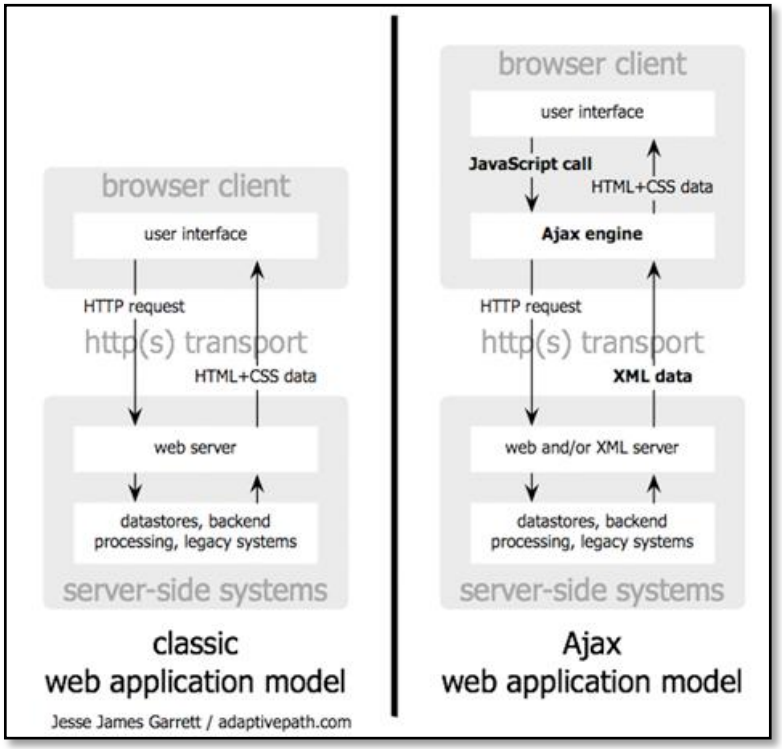


Abbildung 3-1: Vergleich klassische Webapplikationen und einer Ajax-Anwendung (Garrett 2005)

3.1.1.2 Abonnieren mit Web Feed-Technologien – RSS und Atom

Laut Tim O'Reilly (2005) ist RSS eine der Kerntechnologien, die das Web fundamental verändert haben. Die Abkürzung RSS kann je nach Version des Standards für „Really Simple Syndication", „Rich Site Summary" oder „RDF Site Summary" stehen. Die erste RSS-Version (RSS 0.90) stammt aus dem Jahr 1999 und wurde von Netscape[12] für eine personalisierbare Nachrichtenseite („My Netscape Network") entwickelt. (Alby 2008, S. 148) Seitdem hat sich die Entwicklung in unterschiedliche Zweige aufgeteilt, in denen unabhängig voneinander neue Varianten des Standards ausgearbeitet wurden. (vgl. Winer 2003) Die aktuellste und am weitesten verbreitete Version ist RSS 2.0. Dabei handelt es sich – unabhängig von der Version – immer um ein auf XML (Extensible Markup Language) basierendes Format zum plattformunabhängigen Austausch von Webinhalten.

O'Reilly schreibt RSS eine so große Bedeutung zu, weil die sogenannte Web Feed-Technologie zu diesem Zeitpunkt ein Novum darstellt. Mit RSS werden Webseiten nicht einfach nur starr verlinkt, sondern abonniert. Findet eine Aktualisierung der Seite statt, so werden die Abonnenten darüber automatisch informiert ohne die Webseite zuvor aufrufen zu müssen.

Die meist über das HTTP-Protokoll bereitgestellten RSS-Dateien werden als Feeds bezeichnet. Da RSS-Dokumente auf der Metasprache XML basieren, sind sie logisch strukturiert. (siehe Abbildung 3-2) Außerdem unterscheidet sich XML zu anderen Markup-Sprachen, wie HTML, da in XML-Dateien nur der darzustellende Inhalt beschrieben wird, aber nicht wie dieser präsentiert werden soll. Dies hat den Vorteil, dass die ausgetauschten Inhalte in einer beliebigen Oberfläche eingebunden werden können. RSS-Dateien haben keine einheitlichen Dateiendungen. Zu den möglichen Endungen gehören „rss", „xml" und „rdf".

Zum Einlesen und Anzeigen von abonnierten Feeds kommen sogenannte Feedreader oder Aggregatoren zum Einsatz. Diese funktionieren nach dem Pull-

[12] Netscape Communications war ein amerikanisches Software-Unternehmen, was unter anderem den Webbrowser Netscape Navigator entwickelte.

Prinzip, mit dem Sie in periodischen Abständen Feeds nach neuen Inhalten durchsuchen und diese vom Server laden. Die Bezugsquellen sind dabei vom Empfänger frei wählbar und können einfach wieder abbestellt werden. Der Begriff Feed hat sich inzwischen unabhängig von RSS für die einfache Bereitstellung von Inhalten etabliert. Exemplarisch können dafür sogenannte ical-Feeds zum Verteilen von Kalenderdaten genannt werden.

Nicht alle RSS-Versionen sind kompatibel zueinander, bieten dafür aber unterschiedliche Vorteile. Aus dieser Erkenntnis heraus wurde von der Atom Enabled Alliance[13] mit Atom ein weiteres Feed-Format entwickelt, das die Stärken der einzelnen RSS-Versionen vereinen soll. (Alby 2008, S. 148) RSS Version 2.0 und Atom sind die am meisten verbreiten Feed-Formate und werden von den meisten Feedreadern unterstützt. Deshalb sollten Social Software-Anwendungen mindestens eines der beiden Formate anbieten.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rss version="2.0">
  <channel>
    <title>…</title>
    <link>…</link>
    <description>Kurze …</description>
    <language>…</language>
    <copyright>…</copyright>
    <pubDate>…</pubDate>
    <image>
      <url>…</url>
      <title>…</title>
      <link>URL…</link>
    </image>
    <item>
      <title>…</title>
      <description>…</description>
      <link>…</link>
      <author>…</author>
      <guid>…</guid>
      <pubDate>…</pubDate>
    </item>
    <item>...</item>
  </channel>
</rss>
```

Abbildung 3-2: Grundgerüst einer RSS 2.0-Datei

[13] Atom Enabled Alliance ist ein Industrieverband, der sich mit der Veröffentlichung von personengebunden Inhalten und Syndikationswerkzeugen und -diensten beschäftigt.

3.1.2 Allgemeine anwendungsübergreifende Anforderungen

Nachstehend werden weitere wesentliche anwendungsübergreifenden Vorrausetzungen zusammengefasst, die ein Produkt erfüllen muss, um den Anspruch einer vollwertigen Social Software-Kollaborationsplattform gerecht zu werden.

3.1.2.1 Einrichtung und Verwaltung von Kollaborationsbereichen

Wie bereits erwähnt, werden Social Software-Kollaborationsplattformen in einem produktiven Umfeld eingesetzt. Dort ist es üblich, dass für eine erfolgreiche Zusammenarbeit Orgeanisationseinheiten, wie z.B. Arbeitesbereiche oder Projektteams, gebildet werden. Zusätzlich besteht der Trend, dass sich ohne weitere organisatorische Maßnahmen Interessensgruppen bilden. Im Kontext von CSCW werden solche Konstellationen als soziale Entitäten bezeichnet. (vgl. Gross und Koch 2007, S. 16 ff) Damit diese unter dem Dach einer Social Software-Kollaborationsplattform zusammenarbeiten können, sollten eigene Kollaborationsbereiche erstellt werden können.

Neben der Erstellung von Bereichen ist deren Verwaltung ebenfalls ein Wichtiger Aspekt. So sollten Funktionen für ein umfangreiches Rechtemanagement bereitgestellt werden. Es muss also einstellbar sein, Wer Wo Zugriff hat. So wäre es denkbar ungünstig, wenn alle Mitarbeiter eines Unternehmens Zugriff auf den Kollaborationsbereich der Geschäftsleitung hätten. Aufbauend auf diese Anforderung sollte regulierbar sein, welche Rechte ein Nutzer innerhalb eines Bereiches hat. Dass heißt es muss geklärt sein, ob ein Benutzer nur Leser ist, ob er zusätzlich Inhalte erstellen darf oder sogar einen Bereich verwalten kann.

Zur Verwaltung eines Bereiches zählt auch die Abstimmung der Integration der verschiedenen Social Software-Anwendungen, also welche Anwendungen in einem Bereich angezeigt und welche Funktionen für einen Bereich freigeschaltet werden.

Wie aus der CSCW-Forschung bekannt, ist die zyklische, kontinuierliche und evolutionäre Entwicklung ein ausschlaggebendes Erfolgskriterium für den Erfolg von

Groupware. Diese Anforderung ist genauso auf eine Social Software-Kollaborationsplattform übertragbar. Eine bestehende Plattform sollte ohne größeren administrativen Aufwand und zu jeder Zeit um neue Kollaborationsbereiche erweitert werden können. Eine hierarische Anordnung von Bereichen sollte umsetzbar sein, damit diese an bestehende Strukturen angepasst werden können.

3.1.2.2 WYSIWYG-Editoren

WYSIWYG ist ein Akronym, das für das Prinzip „What You See Is What You Get" (dt. Was du siehst, ist, was du bekommst) steht. (vgl. Alby 2008, S. 252) Beiträge, die mit einem WYSIWYG-Editor erzeugt werden, werden bereits bei der Erstellung so angezeigt, wie sie später im Browser für alle Benutzer dargestellt werden. Das ist der optimale Fall, denn letztendlich ist die Anzeige einer HTML-Seite vom Browser abhängig. Ein guter Editor ist auf möglichst viele Browser angepasst, so dass diverse Abweichungen gering gehalten werden können.

Bei Weblogging-Anwendungen ist die Verwendung von WYSIWYG-Editoren bereits Standard.[14] Im Gegensatz dazu ist bei Wikis meist erst das Erlernen einer Markup Language zur Bearbeitung von Inhalten notwendig. Enterprise 2.0 Fallstudien, wie die von Koch und Richter (2009, S. 121), haben gezeigt, dass falls bei der Einführung von Social Software-Anwendungen im Unternehmen WYSIWYG-Editoren verwendet werden, diese Editoren von den Nutzern besser angenommen werden. Dies kann u.a. damit begründet werden, dass nicht alle Benutzer IT-affin genug sind, eine Markup Language zu verwenden.

Grundlage für einen guten Editor ist die Unterstützung von Formatvorlagen, da gerade in Unternehmen die Einhaltung eines Corporate Design[15] ein wichtiges Thema ist. Dafür werden meist zentrale Vorlagen in Form von Cascading Style Sheets festgelegt, die dann als Formatierung für Absätze, Überschriften, etc. auswählbar sind. Daneben sollten noch weitere einfache und individuelle Textformatierungen mög-

[14] Vgl. http://www.blogger.com und http://wordpress.com/, verfügbar am 30.03.2010
[15] Corporate Design sind Gestaltungsvorschriften für das Erscheinungsbild eines Unternehms oder einer Organisation

lich sein (fett, kursiv, unterstrichen, etc.). Außerdem ist die leichte Erstellung von Listen, Tabellen und Links von Vorteil. Zusätzlich erleichtert eine „rückgängig"-Funktion und eine Rechtschreibprüfung die Bearbeitung von Texten. Auch das Einbinden und direkte Hochladen von Bildern im Editor ist für eine komfortable Verwendung sinnvoll.

Die Bereitstellung von intuitiv zu bedienenden WYSIWYG-Editoren ist von enormer Wichtigkeit. Denn nur wenn den Benutzern ein optimales Verhältnis zwischen Aufwand und Nutzen ersichtlich ist, wird diese Software entsprechend gut angenommen.

3.1.2.3 Mehrsprachigkeit

Ein allgemein mögliches Szenario für den Einsatz einer Social Software-Kollaborationsplattform ist ein internationales Unternehmen, in dem die Mitarbeiter nicht nur *eine* Sprache sprechen. In einem solchen Fall wäre es fatal, wenn es nur eine einsprachige Bedienungsoberfläche gibt. Vor diesem Hintergrund sollten Funktionalitäten für eine mehrsprachig geführte Plattform angeboten werden. Dazu gehört, wie bereits angedeutet, die Unterstützung einer mehrsprachigen Benutzerschnittstelle. So sollte zentral oder vom Nutzer selbst einstellbar sein, in welcher Sprache die Menü-Elemente angezeigt werden.

Ferner ist auch die Mehrsprachigkeit bei der Erstellung und Verwaltung von benutzergenerierten Inhalten in Social Software-Anwendungen ein großes Thema. (vgl. Koch und Richter 2009, S. 183 ff) Zum Beispiel können Wiki-Artikel in unterschiedlichen Sprachen verwaltet werden. Dies ist aber eher eine Design-Entscheidung, denn genauso gut kann jeweils ein eigenständiges Wiki für unterschiedliche Sprachen erstellt werden. Deshalb sollte dieser Funktionalität eine nicht zu große Bedeutung zugeschrieben werden.

3.1.2.4 Globale Suchfunktion

Durch Unterteilung in Kollaborationsbereiche und Integration unterschiedlicher Anwendungen liegen die Inhalte in einer Social Software-Kollaborationsplattform verstreut vor. Umso wichtiger ist es, dass diese mit einer gut funktionierenden globalen Volltextsuch-Funktion über alle Anwendungen hinweg durchsucht werden können.

Umso mehr Inhalte vorhanden sind, umso bedeutender wird die Verwendung von Filtern. Sie sollten die Auswahl eines Zeitraums, der Art des Inhalts und den Ort der Suche beinhalten. Die angezeigten Ergebnisse sollten entsprechend nach Relevanz und Erstellungsdatum sortiert werden können.

3.1.3 Usability

Das Wort Usability setzt sich aus den englischen Worten „to use“ (dt. benutzen) und „ability“ (dt. Fähigkeit) zusammen und kann wörtlich mit „Brauchbarkeit“ oder „Bedienbarkeit“ übersetzt werden. Eine wesentlich bessere Übersetzung kann in der deutschen Fassung der EN ISO 9241 („Ergonomie der Mensch-System-Interaktion“) mit Gebrauchstauglichkeit gefunden werden. Darüber hinaus wird im Teil 11 der Norm folgende Definition vorgeschlagen:

„Gebrauchstauglichkeit ist das Ausmaß, in dem ein Produkt von bestimmten Benutzern - der anvisierten Zielgruppe - in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen.“ (ISO 9241 Teil 11)

Ergonomie und Usability sind zwar für alle Software-Systeme gleichbedeutend wichtig, sind aber meist sehr subjektiv und nur kompliziert bewertbar. Eine in diesem Sinne durchzuführende Evaluation würde den Rahmen dieser Diplomarbeit sprengen und könnte vielmehr Stoff für eine eigenständige Arbeit sein.

3.2 Anwendungsspezifische Anforderungen

Social Software-Kollaborationsplattformen zeichnen sich, wie bereits ausgeführt, dadurch aus, dass sie ein breites Spektrum an Social Software-Anwendungen bzw. -Diensten bereitstellen. In den meisten Einsatzszenarien werden zwar nicht alle Anwendungen benötigt, jedoch sollte eine Kollaborationsplattform ein Bündel an ausgewählten, grundlegenden Social Software-Anwendungen mitbringen. So unterschiedlich die Funktionalität der Anwendungen und Dienste ist, so verschieden sind die jeweiligen Vorrausetzungen.

3.2.1 Blogs

Das Wort Blog ist die Kurzform von Weblog, was wiederum ein Kunstwort ist, das sich aus den Begriffen Web und Log (von Logbuch) zusammensetzt. (vgl. Alpar 2008, S. 19) Ein klassisches Weblog ist eine Webseite, auf der ein oder mehrere Autoren in regelmäßigen Abständen Beiträge veröffentlichen, die in chronologisch absteigender Form angezeigt werden.

Ursprünglich wurden Weblogs von Internetnutzern mit HTML-Kenntnissen erstellt, die diese ähnlich einem öffentlichen Tagebuch oder Journal verwendet haben. (vgl. Blood 2000) Mit der Verfügbarkeit von einfach zu bedienenden Bloganwendungen, wie z.B. Blogger und WordPress, ist es inzwischen für die Allgemeinheit der Internetnutzer eine Leichtigkeit ein Weblog zu veröffentlichen. Betrachtet man die Anwendungen technisch, so handelt es sich bei diesen um einfache Content Management-Systeme (CMS).

Laut der Weblogsuchmaschine Technorati[16] gibt es aktuell weit mehr als 130 Millionen Blogs und es werden täglich über 900.000 Blogeinträge veröffentlicht. (vgl. Winn 2009) Diese Zahlen sind schwer überprüfbar, verdeutlichen aber sehr ein-

[16] Vgl. http://www.technorati.com/ - Technorati isteine der größten speziell für Weblogs entwickelten Suchmaschinen.

drucksvoll, dass Weblogs zu den bedeutendsten Webanwendungen überhaupt zählen.

Mit der schieren Anzahl von Weblogs variiert auch die Form der Nutzung. Im Unternehmen können Blogs, je nachdem wie sie verwendet werden, unterschiedliche Stärken aufweisen. Zum einen kann ein Blog das Sprachrohr seines Autors sein und damit eine direkte Einwirkung auf dessen Reputation (Identitätsmanagement) haben. Zum anderen werden in Blogs Informationen zusammengetragen (Informationsmanagement), die in Kommentaren teils kontrovers diskutiert werden (Kommunikationsmanagement). Das alles hat zur Folge, dass Expertenwissen einfacher auffindbar ist und so besser in die tägliche Arbeit einbezogen werden kann.

Eine gute Blogging-Software, die diese Stärken innerhalb einer Social Software-Kollaborationsplattform ausspielen kann, sollte folgende Funktionalitäten unterstützen.

Einträge: Die von den Blogautoren veröffentlichten Artikel werden als Einträge oder auch Postings (bzw. Posts) bezeichnet. Zusätzlich zum Eintragstext sollten diese üblicherweise einen Eintragstitel und das Eintragsdatum besitzen. In einigen Fällen können Blogeinträge bewertet werden, was für eine Qualitätsbestimmung vorteilhaft ist.

Kommentare: Die Leser eines Weblogs haben die Möglichkeit auf Blogeinträge zu reagieren, indem sie meist unmittelbar unter einem Beitrag zu diesem ein Kommentar hinterlassen. So können Diskussionen zwischen den Lesern entstehen, an denen die Autoren ebenfalls teilhaben können. Eine Kommentarfunktion sollte für eine sinnvolle Anwendung vorausgesetzt werden. Die Kommentare sollten zur Übersichtlichkeit chronologisch und mit Darstellung des Datums gelistet werden. Dadurch hat sich auch eine „Antwort-Funktion“ für Kommentare durchgesetzt. Diese ermöglicht das direkte Antworten auf einen Kommentar und zeigt Dialoge formal gesondert an. Daneben integrieren viele Bloganwendungen eine Bewertungsfunktion für Kommentare. So können bei längeren Diskussionen informative Anmerkungen besonders hervorgehoben werden.

Verlinkung: Blogs bieten zweierlei Funktionen zum Vernetzen mit anderen Blogs an. Zum einen das lose Verlinken zu anderen Blogs durch sogenannte Blogrolls, zum anderen können Blogs bidirektional untereinander über Trackbacks und Pingbacks vernetzt werden. Die Grundlage dafür sind dauerhafte Indentifikatoren für erstellte Blogeinträge, die Permalinks genannt werden. Die im Internet so vernetzten Weblogs werden in ihrer Gesamtheit als Blogosphäre bezeichnet. Die Unterstützung dieser Funktionen führt dazu, dass innerhalb der Kollaborationsplattform über alle Kollaborationsbereiche hinweg eine eigenständige Blogosphäre entsteht. Dies hat zum Vorteil, dass ein besserer Überblick über das Geschehen und die Themen der verschiedenen Bereiche vorherrscht.

- **Blogroll:** Um auf thematisch verwandte bzw. interessante Blogs hinzuweisen sollte eine Linksammlung - die Blogroll - angelegt werden können. Darüber hinaus bietet es sich an, dass diese Verweise in Themengebiete unterteilt werden können. Ein zusätzliches System, was zwischen externen und internen Links unterscheidet, wäre sinnvoll.

- **Trackbacks bzw. Pingbacks:** Mit Hilfe eines Rückverweises (engl. linkback) kann angezeigt werden, wann und wo ein Beitrag verlinkt wurde. Diese Funktion wird genutzt, wenn in einem Blogeintrag der Inhalt eines anderen Blogs zitiert oder thematisiert wird. Die Verweise werden als Trackbacks bezeichnet. Sie werden in der Regel unterhalb des Originaleintrags automatisch von der Bloganwendung mit Überschrift und ein Zitat vom referenzierenden Eintrag angefügt. Um die Übersicht zu wahren, sollten die Verweise auf jeden Fall sichtbar getrennt von den Kommentaren gelistet werden. Bevor eine Bloganwendung einen Rückverweis eintragen kann, muss diese zuvor vom bezugnehmenden Blog benachrichtigt werden. Dazu wird bei einem Blogbeitrag immer eine „Trackback-URL“ angegeben. Der Ersteller des verweisenden Blogs muss diese URL (Uniform Resource Locator) selbst ermitteln und beim Erstellen des neuen Beitrags der Bloganwendung übergeben, diese kann dann der Anwendung des Originalbeitrags über die HTTP POST-Methode eine Benachrichtigung schicken. Eine Alternative dazu stellen sogenannte Pingbacks dar. Bei diesen handelt es

sich um eine Art „automatische Trackbacks“, bei denen der thematisierte Beitrag aktiv und automatisch informiert wird, ohne dass der Ersteller dazu eine Trackback-URL benötigt. Dafür wird im Eintragstext des bezugnehmenden Blogs einfach ein Link zum Originalblogeintrag angegeben. Die Bloganwendung erkennt diesen und schickt über eine XML-RPC-Methode eine als Ping bezeichnete Nachricht zum anderen Blog. Am meisten Sinn macht diese Funktion im Internet, wo eine unüberschaubare Menge an Blogs existiert. Deswegen sollten die Verweise als verschmerzbar betrachtet werden.

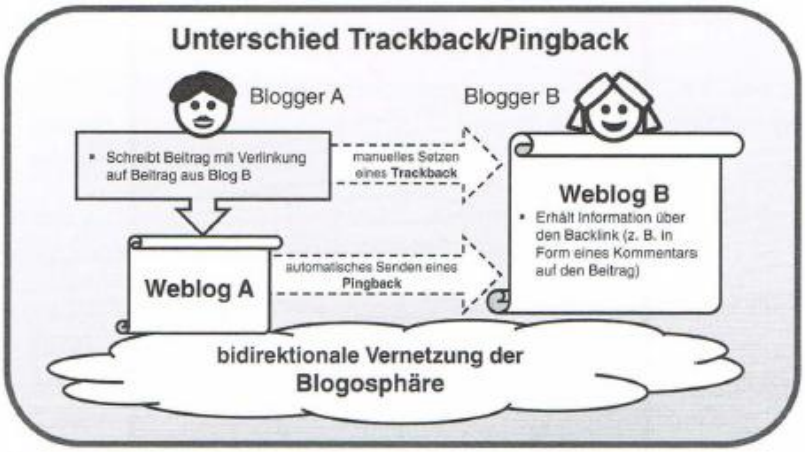


Abbildung 3-3: Unterschied zwischen Trackback und Pingback (Schönefeld 2009, S. 60)

- **Permalinks:** Weblogsysteme stellen zu Beiträgen und zum Teil zu Kommentaren fest vergebene, unveränderbare Webadressen (URL). Diese Links sollten von der Bloganwendung automatisch angelegt werden.

Archiv: Es ist nicht unüblich, dass in einem Blog über einen längeren Zeitraum hinweg regelmäßig Beiträge veröffentlicht werden. Das macht eine Auflistung aller Einträge schnell unübersichtlich. Deswegen wird auf einem Blog nur eine bestimmte Anzahl von Beiträgen angezeigt. Auf der Startseite sind dementsprechend die aktuellsten Einträge gelistet. Über einen Link können sich die Leser zu älteren Beiträgen durchblättern. Dies kann dazu führen, dass die Nutzer bei der Suche nach einem Eintrag eines bestimmten Datums längere Zeit brauchen können. Dieses

Problem wird einfach umgangen, indem eine Linkliste - meist Archiv genannt - mit Verweisen zu allen Einträgen eines Jahres oder eines bestimmten Monats dargestellt wird. Darüber hinaus gibt es Blogs, die über eine Kalenderfunktion verfügen, bei denen Einträge eines bestimmten Tags ausgewählt werden können.

Kategorien und Tags: Ein weitere Möglichkeit, mit denen Bloganwendungen einer drohende Unübersichtlichkeit mit zunehmender Beitragsanzahl entgegenwirken, ist die Taxonomie. Bei dieser findet vom Beitragsersteller eine Klassifikation des Blogeintrages statt. Eine Variante ist dabei die Verwaltung von Kategorien. Es wird eine Liste von Themengebieten festgelegt, die als eine Linksammlung angezeigt werden. Neue Blogeinträge werden entsprechend einer Kategorie zugewiesen. Die Alternative dazu ist das Verteilen von Schlagworten, die als Tags bezeichnet werden. Bei der Erstellung eines neuen Eintrags werden diesem die Schlagwörter zugeteilt und am Ende des Beitrags angezeigt. Mit diesem Vorgehen können Themengebiete einfacher erschlossen werden. Nützlich ist in diesen Zusammenhang auch die Darstellung einer Tag Cloud. Dies ist eine Wortwolke, die alle Schlagwörter und deren Häufigkeit präsentiert, indem öfters auftretende Begriffe grafisch hervorgehoben werden (z.B. größere Schriftart).

Abonnement: Neben dem Abonnieren eines Blogs und der daraus resultierenden Benachrichtigung bei neuen Einträgen sollten auch einzelne Blogbeiträge verfolgbar sein. So können sich die Leser über Veränderungen an einem Eintrag und neuen Kommentaren auf den aktuellen Stand halten.

Suchfunktion: Mit einer Volltextsuche können Leser Blogeinträge nach Begriffen oder Begriffsgruppen durchsuchen. Neben der Suche in einem Blog, ist eine Suchfunktion für das Auffinden innerhalb der Blogosphäre sinnvoll.

Rollenmanagement: Wie bereits angedeutet, funktionieren Bloganwendungen so ähnlich wie Content Management-Systeme (CMS). Zumal einige der Anwendungen den Web Content Management-Systemen (WCMS) zugeordnet werden können. Bei diesen Anwendungen gibt es oft eine weitreichende Rollenvergabe. Für einen Blog

sollten die Rollen: der Autor (Ersteller von Beiträgen), der Editor (Freigabe von Einträgen) und der Leser (Erstellen von Kommentaren) unterschieden werden.

Verwaltung: Blogs sollten auf jeden Fall für die jeweils einzelnen Kollaborationsbereiche erstellbar sein. Darüber hinaus kann das individuelle Einrichten von Blogs je unterschiedlichen Benutzer durchaus praktikabel sein.

3.2.2 Microblogs

Neben dem klassischen Weblog gibt es noch verschiedene Sonderformen, die im Allgemeinen aber unbedeutend sind. Hierbei bilden die sogenannten Microblogs eine Ausnahme. In der Literatur werden diese auch eingedeutscht als Mikroblogs bezeichnet. (vgl. Schönefeld 2009, S. 77) Bekanntestes Beispiel für diesen Anwendungstyp ist Twitter[17], auf dem immerhin mehr als 50 Millionen Beiträge (auch Tweets genannt) pro Tag veröffentlicht werden. (vgl. Weil 2009)

Bei Microblogs werden, wie bei normalen Weblogs, Einträge erstellt, die chronologisch absteigend angezeigt werden. Hiermit sind Gemeinsamkeiten der beiden Anwendungstypen allerdings bereits erschöpft, denn gerade bei Inhalt und Form der Einträge gibt es starke Unterschiede. So handelt es sich bei den Beiträgen in einem Microblog vielmehr um kurze, längenbegrenzte Textnachrichten in denen zeitnahe Status-Updates oder Berichte veröffentlicht werden. Durch die Längenrestriktion der Nachrichten wird zum einen garantiert, dass die Aufmerksamkeit für den Leser zum Auffassen der Information ausreicht, zum anderen ist es für Autoren dadurch um ein vielfaches einfacher, schneller Inhalte zu veröffentlichen.

Die Vernetzung der Microblogs im Internet verfolgt einen anderen Ansatz als Weblogs, denn hier steht vielmehr die Vernetzung der Nutzer im Vordergrund, welche durch das abonnieren von Microblogs anderer Nutzer geschieht. Die Abonnenten eines Nutzers werden als sogenannte Follower (dt. Folger) bezeichnet.

[17] Vgl. http://www.twitter.com – Ist laut Alexa Internet, Inc. in der Liste der meist besuchten Seiten der Welt, auf Platz 12 geführt.

Im Firmenumfeld eignen sich Microblogging-Anwendungen besonders für den Austausch von Awareness-Informationen. Gerade autark handelnde Personen, die aber ein gemeinsames Ziel verfolgen, können dank solcher Anwendungen besser zusammenfinden. Außerdem ist ein Microblog, genau wie ein Weblog auch das Sprachrohr seines Nutzers (Identitätsmanagement). Die Kommunikation der Nutzer untereinander ist möglich, beschränkt sich aber meist auf kurze Dialoge (Kommunikationsmanagement). Ferner kann ein Microblog in einem Unternehmen auch als Feedbackkanal betrachtet werden, über den die aktuelle Stimmung und Geschehnisse verfolgt werden können.

Folgende Anforderungen sollte eine Microblogging-Software innerhalb eine Social Software-Kollaborationsplattform erfüllen:

Einträge: Microblog-Einträge sind zeichenbegrenzt, die Länge für eine Nachricht beträgt, orientiert an der Dimension einer SMS (Short Message Service), üblicherweise zwischen 140 und 160 Zeichen. Durch die Begrenzung der Einträge werden, die Nutzer gezwungen sich nur auf relevante Informationen zu beschränken. Durch diesen Einschnitt erweist sich jedoch das Verlinken von URLs als problematisch, da diese mitunter durch Suchmaschinenoptimierung bereits extrem lang sind. In der Praxis wird dieses Problem durch sogenannte Kurz-URL-Dienste (vgl. Elzer 2009) umgangen. Diese bieten eine möglichst kurze URL an, die nach deren Aufruf zur eigentlichen Adresse weiterleitet. Zusätzlich von Vorteil kann ein Bilder- bzw. Dateianhang sein, der das direkte Einfügen der Inhalte in einen Eintrag erlaubt.

Vernetzung: Primär sollte das gezielte Folgen von Microblogs anderer Nutzer möglich sein. Sinnvoll ist es, wenn dabei auf bereits vorliegende Vernetzungen, wie Freundes- bzw. Kollegenlisten[18], zurückgegriffen werden kann. Weiter sollte ein System integriert sein, was das Folgen von existierenden Gruppen ermöglicht. Dabei können auch erstellte Kollaborationsbereiche einbezogen werden.

Kommunikation: Mit der sogenannten „reply“-Funktion ist den Benutzern eine Möglichkeit gegeben, gezielt einer gewählten Person einen Microblog-Eintrag zu

[18] Siehe dazu Kapitel 3.2.4 Social Networking

senden. Dafür wird in der Nachricht ein „@“-Zeichen und der Nutzername der gewünschten Person eingetragen. Diese Mitteilungen sind im Gegenteil zu anderen Nachrichtensystemen für alle sichtbar. In diesem Zusammenhang bietet es sich an Diskussionspfade zwischen den Nutzern hervorzuheben. Außerdem könnte dieses System so erweitert sein, dass mit dem „@“-Zeichen auch Nachrichten an Gruppen oder Kollaborationsbereichen geschickt werden können.

Suche: Zunächst sollte eine Volltextsuche für das ganze Microblogging-System möglich sein. Die Suche sollte aber auch auf Microblogs einer Gruppe, eines Bereichs oder auf eine Person eingeschränkt werden können.

Tagging: Ein etabliertes System für das Verschlagworten von Microblog-Beiträgen sind die sogenannten Hashtags. Dabei werden alle Begriffe innerhalb eines Eintrags, die mit dem „#“-Zeichen beginnen, vom System als Tags erkannt. Die entsprechenden Beiträge, die zu einen Tag gehören, können wie üblich mit einer Tag Cloud aufgerufen werden. Alternativ ist eine Auflistung von Trends, also derzeitig oft verwendeten Tags denkbar.

Schnittstellen: Eine Stärke von Microblogging-Anwendungen wie Twitter ist der allgegenwärtige Zugriff auf die Anwendung. So können die Tweets per Handy über einen mobilen Client oder SMS versendet werden. Beim Einsatz in einem Intranet könnte dies jedoch durch Abschirmung schwierig sein. Deswegen sollte auch das Versenden über E-Mail in Betracht gezogen werden.

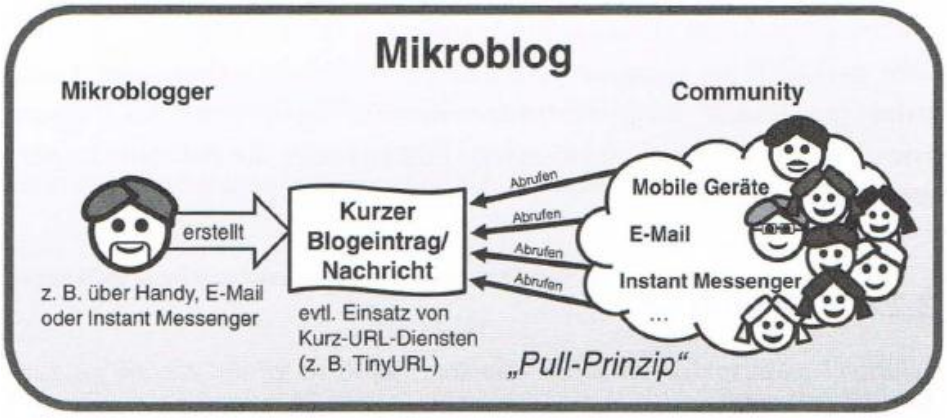


Abbildung 3-4: Microblog (Schönefeld 2009, S. 78)

Verwaltung: Wie bei Blogs sollten mehrere Microblogs in den Kollaborationsbereichen zu unterschiedlichen Themen einzurichten sein.

3.2.3 Wikis

Wiki-Software, teils auch Wiki-Engines genannt, sind einfache, meist webbasierte Content-Management-Systeme, deren Programmatik die schnelle und unkomplizierte Bereitstellung und Bearbeitung von Inhalten ist. Aus diesem Grund wurde auch der hawaiische Begriff „Wiki“, der so viel wie „schnell“ bedeutet, als Bezeichnung gewählt. (vgl. Ebersbach 2008, S. 14 ff) Die mit Hilfe dieser Anwendungen erstellten Wikis sind Sammlungen von miteinander vernetzten Dokumenten und Dateien, die nicht nur online im Webbrowser angezeigt, sondern auch von den Nutzern editiert werden können. (vgl. Klobas 2006, S. 3)

Trotz ähnlichem Ansatz unterscheiden sich Wiki- und Bloganwendungen fundamental in ihren Funktionsweisen. Weblogs sind primär für eine one-to-many-Kommunaktion konzipiert, dass heißt es gibt in der Regel nur einen Autor je Beitrag. Dagegen findet bei Wikis eine many-to-many-Kommunikation statt, bei der eine Gruppe von Nutzern ihr Fachwissen zu bestimmten Themen zusammenträgt und gemeinsam dafür Artikel erstellt und versionsgesichert bearbeitet. Außerdem ist das chronologische Anordnen von Beiträgen ein elementares Merkmal von Blogs, hingegen sind Wikis auf den Inhalt fokussiert. (vgl. Raabe 2007, S. 40 ff)

Wikis unterstützen in erster Linie das Informationsmanagement, so werden diese oft als Wissensspeicher und Dokumentationsplattform in Unternehmen genutzt. Zudem stellen gute Wiki-Engines Funktionalitäten zur Verfügung mit denen die Einträge diskutiert und Veränderungen nachvollzogen werden können. Somit wird von diesen Anwendungen in einem beschränkten Ausmaß Identitätsmanagement sowie Kommunikation ermöglicht.

Die in die Social Software-Kollaborationsplattform eingebundene Wiki-Software sollte, wie folgt aufgebaut sein:

Artikel: Die in einem Wiki veröffentlichten Inhalte sind hauptsächlich Text, können aber genauso multimediale Inhalte wie Bilder und Videos beinhalten. Für die Erstellung von Inhalt wird in der Regel eine einfach zu erlernender Wiki-Syntax angeboten, mit dessen Hilfe die Texte gut lesbar und gegliedert gestaltet werden können. Auf einen WYSIWYG-Editor sollte aus den bereits aufgezeigten Gründen trotzdem nicht verzichtet werden. Außerdem werden oft Gliederungsvorlagen für die Artikel angeboten, um eine einheitliche Beschaffenheit der Beiträge zu garantieren. In vielen Fällen, wie bei einer Projektdokumentation, bietet es sich auch an, dass zusätzliche Dateien direkt an einen Artikel angehangen werden können.

Qualitätssicherung: Eine Schwierigkeit bei der kollektiven Arbeit an einem Artikel ist die Sicherstellung der Korrektheit des Inhalts. Deswegen bieten Wiki-Engines Funktionen für eine Versionskontrolle an. Hier wird angezeigt Wer Wann Welche Veränderungen erledigt hat. Außerdem können mit diesen Funktionen ältere Versionen bei Bedarf wiederhergestellt werden. Zudem können die Inhaltsseiten und deren Bearbeitung in Diskussionsforen oder Kommentaren von den Nutzern besprochen werden. Desweiteren könnte auch eine Funktion bereitgestellt werden, mit der der „Reifegrad“ eines Artikels ablesbar ist.

Übersichtsseite: Auf jeden Fall sollte eine Einstiegsseite erstellbar sein, die zum Beispiel eine Übersicht über den Inhalt, ausgewählte interessante Artikel oder aktuelle Änderungen gibt. Förderlich wäre es, wenn diverse Anzeigen dabei automatisch erstellt werden.

Verlinkung: Die interne Vernetzung der Wiki-Artikel untereinander ist ein wichtiger Bestandteil eines brauchbaren Wikis. Generell wird dies durch eine einfache Wiki-Syntax von den Nutzern realisiert. Die Verlinkung sollte konsistent sein, so sollte beispielsweise von der Software erkannte werden, falls ein Artikel umbenannt wird. Außerdem sollten auch Artikel aus einem anderen Kollaborationsbereich bei Bedarf verlinkt werden können.

Kategorien und Tagging: In einem Wiki können oft Kategorien erstellt und diese Artikeln zugewiesen werden. Nützlich ist es, wenn die Kategorien entsprechend grafisch nachvollziehbar am Ende der Einträge angezeigt werden. Auch die Ver-

wendung von Tags kann eine brauchbare Funktion sein, infolgedessen sollte auch beispielweise eine Tag Cloud auf der Übersichtsseite einzufügen sein.

Abonnement: Auf der einen Seite sollte ein komplettes Wiki abonniert werden können, damit auf neu erstellte Artikel hingewiesen werden kann. Es sollten aber auch einzelne Artikel wahlweise mit einem Abonnement nachverfolgt werden können, damit die Nutzer über aktuelle Veränderungen zu einem bestimmten Thema informiert bleiben können.

Artikel-Export: In einem Wiki wird Wissen gespeichert, was auch an andere Stelle benötigt werden kann. Deswegen sollte eine einfache Exportfunktion für Artikel bereitgestellt werden. Hier sollten gängige Formate verwendet werden oder auch eine Funktion zum Versenden per E-Mail angeboten werden. Wichtig ist, dass, falls es eine Exportfunktion für ein komplettes Wiki existiert, diese deaktivierbar ist. So kann Wissensdiebstahl erschwert werden.

Suche: Neben dem gezielten Durchsuchen eines bestimmten Wikis sollten auch global all Wikis einer Plattform durchsucht werden können. Dabei sollten zunächst Ergebnisse mit ähnlichen Artikelüberschriften und anschließend Resultate nach Volltextsuche aufgelistet werden.

Rollenmanagement: Bei der Rollenverteilung wird zunächst zwischen Editoren, die Inhalte bearbeiten dürfen, und Lesern unterscheiden. Die Leser dürfen die Inhalte nur betrachten und optional diese auch verschlagworten. Eine besondere Rolle stellen „Wiki-Gärtner“ dar, die zur Pflege eines Wikis beauftragt werden. Sie können ganze Artikel löschen, besondere Anmerkungen zum Inhalt eines Eintrags hinterlassen und gegebenenfalls die Einstellungen zum „Reifegrad“ einer Inhaltsseite verändern.

Verwaltung: Es sollte mindestens ein Wiki je Kollaborationsbreich erstellbar sein.

3.2.4 Social Networking Services

Als Social Networking Services (SNS) *„werden Dienste bezeichnet, die ihren Nutzern Funktionen zum Indentitätsmanagement (d.h. zur Darstellung der eigenen Person in der Regel in Form eines Profils) zur Verfügung stellen und darüber hinaus die Vernetzung mit anderen Nutzern (und so die Verwaltung eigener Kontakte und Pflege des Netzwerks) ermöglichen.“* (Richter und Koch 2008, S. 2)

Neben Blogs und Wikis gehören die Social Networking Services und die mit den Diensten erstellten sozialen Netzwerke zu der am meist genutzten Social Software im Internet. So hat Facebook[19], eine der größten sozialen Netzwerkseiten, über 400 Mio. Nutzer weltweit (Facebook 2010) und wird zusätzlich bei Alexa als die am zweit häufigsten besuchte Seite der Welt geführt. (Alexa 2010b)

Der Vorteil von SNS zu anderen Systemen in Unternehmen wie dem Telefonverzeichnis oder Firmenorganigrammen ist, dass bei der Gestaltung die eigentliche Person zu Wort kommt. So wird dieser Platz bei der Darstellung des eigenen Profils und der eigenen Fähigkeiten gelassen. Damit werden die Grundlagen für eine informelle Vernetzung der Mitarbeiter geschaffen, die jenseits der etablierten organisatorischen Strukturen wirken (Abbildung 3-5). (vgl. Schönefeld 2009, S. 70)

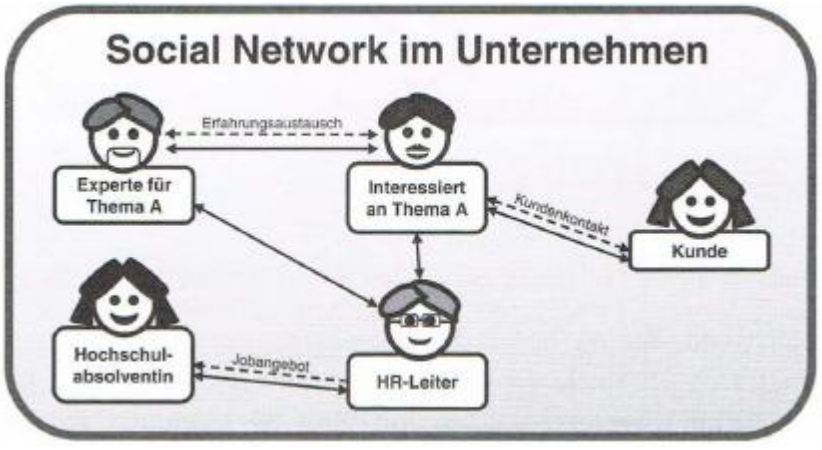


Abbildung 3-5: Soziales Netzwerk im Unternehmen (Schönefeld 2009, S. 70)

[19] http://www.facebook.com, verfügbar 10.04.2010

Die wichtigen Funktionen der Social Network Services, die in einer Social Software Kollaborationsplattform umgesetzt werden sollten, sind:

Profil: Das Kernstück der Social Networking Services ist das Nutzerprofil, dass eine Art personelle Informationsseite ist. Die Nutzer sollten die Daten, die im Profil zu sehen sind, selbst verwalten können. Außerdem sollte es auch möglich sein, dass ausgewählte Informationen: wie z.B. Telefonnummer oder Vorgesetzter des Nutzers, zentral und ohne Zutun der Nutzer eingespeist werden. Zusätzlich sollten bei einer guten Plattform benutzerdefinierte Felder erzeugbar sein, da sich die Anforderungen an ein Profil von Unternehmen zu Unternehmen stark unterscheiden können. Wichtige Informationen die exemplarisch in einem Profil stehen können sind Profilname, Profilbild, Profilbeschreibung, Team, Expertisen bzw. Skills, Kontakte und Projekte an denen der Mitarbeiter arbeitet(e). Außerdem sollen Einstellungen vom Nutzer getätigt werden können, mit denen er auf freiwilliger Basis steuern kann, welche Gruppen Einsicht auf welche Daten haben.

Vernetzung: Wie schon erwähnt, ist die Vernetzung ein wichtiger Bestandteil der Social Networking Services. In einer Social Software-Kollaborationsplattform sollte diese Funktion durch sogenannte Freundes-/Kollegenlisten realisiert werden. Diese Listen sollen zur Stärkung von Nutzerbeziehung dienen. Jeder kann in seinem Profil darstellen, mit welchen Nutzern er engere Kontakte pflegt. Dies kann der Fall sein, weil zusammen ein Projekt bearbeitet wird, der Nutzer als Experte geschätzt wird oder weil an ähnlichen Themen gearbeitet wird. Es bedarf keiner gegenseitigen Rückversicherung. Die Nutzer können also auch einfach aufhören jemandem zu folgen und die Verbindung lösen, wenn zum Beispiel ein gemeinsames Projekt abgeschlossen ist. Das bildet besser den eigentlichen Arbeitsprozess ab – je nachdem woran gerade gearbeitet wird, haben verschiedene Personen unterschiedlich intensiv miteinander Kontakt. Über einen gewissen Zeitraum werden hier Veränderungen eintreten und diese Dynamik sollte ein soziales Netz im Unternehmen auch abbilden. Interessant kann es zudem sein, zu sehen, über welche Mitarbeiter man in Beziehung zu dem Nutzer steht, dessen Profilseite momentan besucht wird. Ein „Social Graph“ ist in diesem Fall eine nützliche Funktion. Er stellt eine Visualisierung – also eine grafische Aufbereitung – des Beziehungsnetzwerkes dar. Es kön-

nen mit dieser Funktion Erkenntnisse zu Verbindungen aufgezeigt werden, die bis dahin verborgen waren.

Activity Stream: Aufbauend auf die Kollegenlisten sollte die Einbindung von sogenannten Activity Streams (dt. Aktivitätsströme) erfolgen. So wird die Kollegenliste ähnlich wie bei der Follow-Funktion von Microblogs zu einer chronologischen Anzeige der letzten Aktivitäten und Arbeiten der Kontakte innerhalb Social Software-Kollaborationsplattform. Dazu gehören auch Infos, welche Dokumente erstellt, welche Wikiseiten geändert, welche Kommentare geschrieben wurden. Diese Art der Vernetzung macht nur Sinn, wenn das Soziale Netzwerk keine alleinstehende Plattform ist, sondern eine enge Integration mit allen Anwendungen besteht, auf denen tatsächlich zusammengearbeitet wird. Im Weiteren sollten die Nutzer steuern können welchen Aktivitäten sie folgen und welchen nicht.

Kommunikation: Es bietet sich an, wenn an das Profil ein System zum Austausch von privaten Nachrichten angebunden ist, über die Nutzer direkt kommunizieren können. Zusätzlich sind bei Sozial Netzwerken meist Gästebücher auf den Profiilseiten zum Hinterlassen kurzer öffentlicher Nachrichten. Der Profilbesitzer sollte die Nachrichten löschen und darüber hinaus einstellen können, wer die Nachrichten einsehen kann.

Konsistente Identitätsverfolgung: In einer Social Software-Kollaborationsplattform ist eine durchgängige einheitliche Identität der Nutzer, die von allen Anwendungen und Inhalten aus zurückverfolgt werden kann, eine signifikante Eigenschaft. Sie ist fundamental für das Auffinden von Wissensträgern und Kompetenzen und genauso bedeutend für eine Verantwortlichkeitensuche. Demnach sollte beispielsweise direkt vom Kommentar eines Blogeintrags auf das Profil des Erstellers und damit dessen Kenntnisse geschlossen werden können. Ein in diesem Sinne nützliches Merkmal sind so sogenannte „Badges“, die eine virtuelle Visitenkarte darstellen und als Signatur verwendet werden können. Sie können als Vorschau für ein Profil angezeigt werden und sollten als optional betrachtet werden.

Tagging: Für Nutzer ist die Tatsache, dass eine andere Person Teil seines Netzwerks ist, meist weniger interessant als die Themen, die diese Personen verbinden. Deswegen ist es eine nützliche Funktion, wenn Nutzer andere Profile Tags zuweisen können. Damit pflegt das Netzwerk das Profil und macht es nicht nur aktueller und vielschichtiger, sondern auch glaubwürdiger. Natürlich sollte auch bei dieser Funktion die Kontrolle beim Besitzer des Profils legen, sodass Tags von diesem gelöscht werden können.

3.2.5 Social Tagging

Unter Social Tagging (auch Collaborative Tagging oder gemeinschaftliches Indexieren) *„versteht man einen Vorgang, bei dem eine Gemeinschaft von Benutzern eines Systems die Objekte in diesem System mit Metadaten (Schlüsselwörtern und Kategorisierungen) charakterisiert.“* (Back, Gronau und Klaus 2009, S. 39) Bei den bisher vorgestellten Anwendungen – Weblogs und Wikis – ist das primäre Ziel die Erstellung neuer Inhalte. Im Unterschied dazu werden bei Social Tagging Inhalte mit Metainformationen angereichert. Sie stellen in diesem Fall konkret einfache Schlüsselworte dar, die den Inhalt beschreiben bzw. klassifizieren.

Das Konzept des Social Taggings gibt es bereits seit dem Jahr 2003, in dem die ersten Anwendungen mit Tag-Unterstützung im Web veröffentlicht wurden. Jedoch ist Tagging kein komplett neues Konzept, denn in herkömmlichen Wissensorganisationssystemen hat die Sacherschließung eine lange Tradition. (vgl. Heckner 2009, S. 60) Jedoch ordnen im Gegensatz zur klassischen Indexierung die Nutzer (Inhalts-)Objekten frei wählbare Tags zu. Dies hat in erster Linie für die Nutzer ein Eigennutzen, da diese die Objekte so später einfacher wieder auffinden können. Aber auch die Wahl von Tags vereinfacht die Dinge für die Nutzer ungemein, denn eine Einordnung von Objekten in klare (hierarchische) Strukturen ist oft ein komplizierter bis unmöglicher Vorgang. Dagegen fällt es Menschen leichter unterschiedliche Schlüsselworte für Objekte zu vergeben. Zudem können einzelne Nutzer die Tag-Sammlungen der Allgemeinheit zugänglich machen, was besondere Effekte für ein Netzwerk verspricht. So können zum Beispiel Dokumente und deren Ersteller mit

identischen Tags in Verbindung gebracht werden. (vgl. Koch und Richter 2009, S. 46) Die so entstehende Vernetzung der Tags wird auch als Folksonomy, was ein Kofferwort aus „folk taxonomies“ (dt. Laien-Taxonomien) ist, bezeichnet. (vgl. Raabe 2007, S. 42)

Tagging und Tags sind ein wichtiger Bestandteil für das Informationsmanagement einer Social Software-Kollaborationsplattform. Neben der Integration dieser Funktionalitäten in die anderen Anwendungen sollte zusätzlich eine eigenständige Tagging-Anwendung angeboten werden, die folgende Anforderungen erfüllt:

Tagging: Zuerst einmal sollten primär so viel wie möglich unterschiedliche Inhalte mit Tags versehen werden können. Darunter zählen unter anderen die bereits aufgezählten Blogeinträge, Blogkommentare, Wiki-Beiträge, Foren-Beiträge, Benutzer, aber auch statische Seiten/Inhalte. Zusätzlich wäre es gut, wenn Videos und Bilder getagt werden könnten. Bei der Eingabe der Tags sollten verschiedene vorgefertigte Tags zur Auswahl stehen. Ein noch besseres System ist, wenn beim Eintippen dynamisch Vorschläge angezeigt werden. Dadurch wird garantiert, dass nicht zu viele unterschiedliche Tags nebeneinander existieren, die ähnliche Intention haben. Zuzüglich sollen die Nutzer auswählen können, ob die Tags privat oder öffentlich sind. Ergänzend ist ein System sinnvoll bei dem Notizen zum getagten Inhalt angelegt werden können.

Anzeigen von Tags: Die übliche Anzeige der Tags wird über Schlagwortwolken, den sogenannten Tag Clouds realisiert. Es ist natürlich praktisch, wenn diese an so vielen verschiedenen Stellen wie möglich angezeigt werden können, um eine große Variation bei der Nutzung zu garantieren. Zumindest sollte eine Tag Cloud pro Bereich erstellbar sein. Ferner könnte auch eine Schlagwortwolke pro Nutzer erstellt werden, damit andere sehen können, welche Schlüsselworte ein Nutzer vergibt. Sinnvoll ist es, wenn einzelne Tags über die Tagging-Anwendung abonniert werden können, damit sich die Nutzer gezielt zu verschiedenen Themen auf den aktuellen Stand halten können. Außerdem können Benutzer gut erkennen, wer sich mit den gleichen Themen beschäftigt, wenn für andere ersichtlich ist, Wer welchen Tags folgt.

Social Bookmarking: Bei Social Bookmarking-Systemen handelt es sich um Anwendungen zum gemeinschaftlichen Erfassen und Kategorisieren von Weblesezeichen (engl. bookmarks). (vgl. Raabe 2007, S. 42) Im Internet gibt es inzwischen eine Vielzahl dieser Anwendungen. Eine der erfolgreichsten Seiten ist delicious.com[20], welche, wie die meisten Systeme, beim Kategorisieren der Lesezeichen primär auf die Verwendung von Tags setzt. Daran orientiert sollte ein Social Bookmarking-Dienst innerhalb einer Social Software-Kollaborationsplattform stark mit dem bereits vorhandenen Tagging-Dienst verknüpft sein. Die eigentliche Besonderheit bei der Verschlagwortung ist, dass es sich dabei beim ausgezeichneten Inhalt um externe Links, statt um interne Objekte der Plattform handelt. Für die Nutzer bietet ein Social Bookmarking-System im Gegensatz zu fest in einem Web-Browser gespeicherten Lesezeichen klare Vorteile. Zum einen kann von jedem internetfähigen Gerät, welches Zugriff auf die Plattform hat, auf die Sammlung zugegriffen und neue Einträge hinzugefügt werden. Weiterhin besteht für andere Nutzer bei einer Recherche die Option auf diese Liste zuzugreifen, demzufolge können diese schneller geeignete Quellen finden. Mehr sogar kann ein derartiges System einen Beitrag für das Beziehungsmanagement leisten, wenn die Nutzer Bookmarks anderer verfolgen können. So gibt es bei delicious.com sogar eine Funktion bei der neue Lesezeichen direkt anderen Nutzern empfohlen werden können. Parallel könnte auch eine Funktion zum Erstellen von eigenen Linksammlungen für Bereiche bestehen, die alle Benutzer des Bereichs einsehen und bearbeiten können.

3.2.6 Weitere Anwendungen und Dienste

Die bereits aufgelisteten Anwendungen und Dienste sind diejenigen, die unbedingt in eine Social Software-Kollaborationsplattform integriert werden sollten. Darüber hinaus gibt es aber noch weitere unterstützende Hilfsmittel, die auf dem ersten Blick entweder keine typische Social Software oder weniger relevant sind. Deshalb

[20] http://www.delicious.com, verfügbar am 15.04.2010 – Eine seit 2003 existierende Social Bookmarking-Webanwendung, die inzwischen Yahoo gehört.

sollen folgende Programme zwar in eine Bewertung, wenn auch mit geringerer Gewichtung, einbezogen werden:

3.2.6.1 Foren

Diskussionsforen ähneln in ihrer Funktionsweise zwar Weblogs existieren aber schon wesentlich länger. (vgl. Richter und Koch 2009, S. 33) Die Nutzer veröffentlichen in Foren Diskussionsbeiträge (Postings), die von anderen gelesen und beantwortet werden können. Dabei können Gedanken, Meinungen und Erfahrungen zu verschiedenen Themen diskutiert und archiviert werden. Die Beiträge die sich dabei auf das Thema eines eröffnenden Beitrags beziehen, werden als Thread (dt. Faden) oder Topic (dt. Thema) bezeichnet. Diskussionsforen sind im Gegensatz zu Blogs auf den Kooperationsprozess fokussiert; sie machen den Austausch von Meinungen und die Bildung eines gemeinsamen Verständnisses möglich. Das Ergebnis einer Diskussion allerdings bleibt meist implizit in den Beiträgen enthalten und muss nachträglich extrahiert und verdichtet werden.

Am meisten verbreitet im Internet sind die sogenannten (Bulletin) Boards, die auf Grund ihrer ähnlichen Funktionsweise mit einem schwarzen Brett so genannt werden. Die Themen werden meist hierarisch angeordnet. Innerhalb einer Themengruppe werden dann die Threads nach Aktivität angeordnet, wobei immer der Titel des ersten Beitrags angezeigt wird. Die Postings eines Threads werden chronologisch auf einer Seite angezeigt und nach einer gewählten Anzahl von Beiträgen auf eine Folgeseite umgebrochen. Die Nutzer können zu dem meist Threads und deren Überthemen abonnieren. Ferner können eröffnende Beiträge eine Umfrage beinhalten, bei der die Benutzer abstimmen können. So kann eine These unterstrichen oder eine Frage von der Gemeinschaft beantwortet werden. Ein Forum in einer Social Software-Kollaborationsplattform sollte als Bulletin Board umgesetzt werden und je Kollaborationsbereich sollte mindestens ein Forum erstellbar sein.

3.2.6.2 Instant Messaging

Unter Instant Messaging (kurz IM, dt. „sofortiger Nachrichtenversand“) *„versteht man einen Dienst, bei dem die synchrone Kommunikation textuell erfolgt. Die Kommunikationspartner können also Textnachrichten eingeben, die dann sofort dem Kommunikationspartner zugestellt werden. Viele IM-Programme bieten inzwischen auch die Möglichkeit zu Video- oder Telefonkonferenzen an, insofern ist der Übergang zu Sprach- und Videokonferenzen fließend.“* (Richter und Koch 2009, S. 67)

In einigen Publikationen, wie auch der eben zitierten, wird Instant Messaging zumindest im erweiterten Kreis der Social Software-Anwendungen aufgeführt. Andere Veröffentlichungen sehen es lediglich als unterstützendes Werkzeug mit Verweis auf den Aspekt der nicht vorhandenen Sichtbarkeit von Nutzern außerhalb des bereits vorhandenen Netzwerks. (vgl. Raabe 2007, S. 44 ff)

Instant Messenger sind in der Praxis meist eigenständige Programme. Falls diese also nicht direkt in die Plattform integriert sind, sollten zumindest die Profile der Anwendungen verknüpft sein. Dass heißt die gleichen Profil verwendet werden und in der Plattform angezeigt werden, ob ein Nutzer im Messenger verfügbar ist. Außerdem sollten die Nutzer über die Profilseite direkt in die Kontaktliste, die optimaler weise auch vorhandene Gruppen darstellt, des IM-Client eingefügt werden können. Im Weiteren gehören Funktionen wie Statusauswahl und Nachrichtenlog zur Standardausstattung eines Clients.

3.2.6.3 Gruppeneditoren

Gruppeneditoren sind Editoren, die das gemeinsame Editieren von Dokumenten unterstützen. Bei Wikis können zwar auch mehrere Nutzer an einer Webseite arbeiten, jedoch steht bei Gruppeneditoren die synchrone Zusammenarbeit im Vordergrund. (vgl. Richter und Koch 2009, S. 42)

Dank der Unterstützung von Ajax gibt es im Web inzwischen einige Anwendungen die eine synchrone Erstellung und Bearbeitung eines Dokuments ermöglichen. Ein gutes Beispiel dafür ist Google Docs[21]. Nach der Anmeldung können in dieser Webanwendung ohne zusätzliche Installationen direkt im Webbrowser Tabellen und Textdokumente erstellt und andere Nutzer zum Editieren eingeladen werden. Ähnliches ist für eine Social Software-Kollaborationsplattform erstrebend. Ganz wichtig bei der synchronen Zusammenarbeit ist ein Kommunikationskanal, über den eine Koordination der Nutzer stattfinden kann. Deshalb sollte ein Instant Messaging-Dienst mit dem Gruppeneditor verbunden sein. Überdies sollten Funktionen zum Import und Export wichtiger Dateiformate (.xls, .csv, etc.), zum automatischen Speichern während der Arbeit und zur Aufgabeneinteilung zur Verfügung stehen.

3.2.6.4 Collaborative Filtering

Unter dem Begriff Social Navigation werden die Konzepte zusammengefasst, die Benutzern die Möglichkeit geben sich bei ihrer Navigation (Auswahl) am Verhalten und den Hinweisen anderer zu orientieren. (vgl. Richter und Koch 2009, S. 65) Hintergrund dieser Idee ist, dass sich Menschen beim Finden von relevanten Informationen und beim Navigieren soziale Informationen nutzen, denn in solchen Fällen fragen Menschen oft Bekannte nach Empfehlungen oder leiten diese aus den Aktivitäten anderer ab.

Systeme die sich diese Idee in virtuellen Informationsräumen zu Nutze machen, werden als (automatische) Collaborative Filtering-Anwendungen bezeichnet. „*Sie errechnen Ähnlichkeiten zwischen Benutzerprofilen, um aus einer Vielzahl von Bewertungen individuelle Empfehlungen zu generieren.“* (Baier, Weinreich, Wollenweber 2004, S. 3) Aufbauend darauf werden Profile die Gemeinsamkeiten zum anfragenden Nutzerprofil aufweisen mit höherer Priorität bewertet. Ein gutes Beispiel dafür sind die Kaufempfehlungen auf Amazon[22].

[21] http://docs.google.com, verfügbar am 15.04.2010
[22] http://www.amazon.de, verfügbar am 17.04.2010 – Amazon ist einer der größten Internetversandhäuser der Welt.

Bei der Umsetzung in einer Social Software-Kollaborationsplattform sollten Vorschläge zu Inhalten, aber auch zu Personen den Nutzern angezeigt werden. Außerdem wäre es sinnvoll, wenn nicht nur die Profile verglichen werden, sondern auch die Aktivitäten und das Suchverhalten innerhalb der Plattform.

4 Evaluierung

In diesem Kapitel werden drei ausgewählte Produkte vorgestellt, welche anschließend mit Hilfe der bereits im Kapitel 3 erstellten Anforderungen auf die Umsetzung von Social Software-Funktionalitäten untersucht werden. Das Ergebnis dieser Analyse wird daraufhin anhand einer eigens erstellten Bewertungsmatrix dargestellt und ausgewertet.

4.1 Ausgewählte Produkte

Nach einer Vorstellung der selektierten Produkte werden für diese die einzelnen Social Software-Komponenten betrachtet. Beginnend wird der Microsoft SharePoint Server 2010 (kurz SharePoint oder SharePoint Server), der aus bereits aufgeführten Gründen im Mittelpunkt dieser Diplomarbeit stehen soll, vorgestellt. Die beiden Alternativprodukte wurden so gewählt, dass sie sich im Ursprung ihrer Entwicklung unterscheiden. So kann ein Überblick über die derzeitige Entwicklung am Markt vermittelt und gleichzeitig die Integration der betrachteten Anwendungen im SharePoint aus verschiedenen Blickwinkeln eingeordnet werden.

4.1.1 Der Microsoft SharePoint Server 2010

Die aktuellste Version des SharePoint Server wurde am 12. Mai 2010 veröffentlicht (vgl. Rubenstein 2010) und wird von Microsoft selbst als „Business Plattform für Zusammenarbeit im Unternehmen und im Web“ (Microsoft 2010a) bezeichnet. Im Jahr 2009 hat Microsoft bekannt gegeben, dass SharePoint zu den am schnellst wachsenden Produkten in deren Firmengeschichte gehört. So wurden 1,3 Milliarden US-Dollar in einem Jahr erwirtschaftet, was einem Wachstum von 20 Prozent entspricht. (vgl. Microsoft 2009) Man kann davon ausgehen, dass sich diese Erfolgsgeschichte mit der neuesten Version fortsetzt.

4.1.1.1 Allgemeines zum SharePoint Server

Dem SharePoint Server 2010 gehen verschiedene Versionen voraus, wobei Microsoft nicht den aus dem Web bekannten Ansatz der Perpetual Beta verfolgt. Vom amerikanischen Unternehmen werden nur „fertige“ Produkte veröffentlicht, bei denen mit Updates lediglich Fehler beseitigt, aber keine neuen Funktionen hinzugefügt werden.

SharePoint Portal Server 2001 und SharePoint Team Services

Das erste Produkt der SharePoint-Reihe erschien Anfang 2001 und hatte seine Ursprünge als Dokumentenmanagement- und Indexierungsanwendung. Jedoch wurde mit der finalen Version der Fokus auf den zu dieser Zeit rasant wachsenden Markt der Portalanwendungen gelegt. Diese Entscheidung von Microsoft kann auch anhand der Namensgebung nachverfolgt werden, da die finale Version als SharePoint Portal Server 2001 (kurz SPS) veröffentlicht wurde. (vgl. Richardson 2006)

„Ein Portal ist definiert als eine Applikation, welche basierend auf Webtechnologien einen zentralen Zugriff auf personalisierte Inhalte sowie bedarfsgerecht auf Prozesse bereitstellt. Charakterisierend für Portale ist die Verknüpfung und der Datenaustausch zwischen heterogenen Anwendungen über eine Portalplattform.“ (Kirchhof et al. 2004, S. 5)

Technischer Unterbau der Plattform war zu diesem Zeitpunkt die serverseitige Skript-Engine ASP (Akronym für Active Server Pages) und eine spezielle, modulare Version des Microsoft Exchange Servers. (vgl. Liu 2007)

Zusätzlich wurde mit den SharePoint Team Services (kurz STS) ein kostenloses Add-on für Office 2000 angeboten, was eine webbasierte Teamzusammenarbeit ermöglichen sollte. (vgl. Richardson 2006) Die STS waren jedoch mit maximal 75 Nutzern pro Seite stark begrenzt. (vgl. Oleson 2007)

SharePoint Portal Server 2003 und Windows SharePoint Services

Bei der nächsten Version wurden einige fundamentale technologische Veränderungen vorgenommen. So wurde das komplette Frontend durch die damals neue serverseitige Technologie ASP.NET ersetzt und außerdem wurde mit dem Microsoft SQL Server auf ein relationales Datenbankmanagementsystem umgestellt. Dies hatte zur Folge, dass die Skalierbarkeit und Portalfunktionen der Anwendung verbessert, dafür aber einige der Dokumentenmanagement-Funktionen eingespart wurden.

Es wurden die SharePoint Team Services in Windows SharePoint Services (WSS) umbenannt und stärker mit der neuen Ausführung des SharePoint Servers, dem SharePoint Portal Server 2003, zusammengeführt. Dafür wurde der neue SPS aufbauend auf den WSS, welche ein Bestandteil des Windows Server 2003 wurden, erstellt. Die Hauptfunktionen des SPS 2003 waren bei der Indizierung bzw. Suche, Personalisierung und verbessertes Management bzw. Taxonomie. (vgl. Richardson 2006)

Microsoft Office SharePoint Server 2007

Im Jahr 2007 erschien der direkte Vorgänger des Microsoft SharePoint Server 2010, der Microsoft Office SharePoint Server 2007 (kurz MOSS). Er stellt einen großen Entwicklungssprung dar. Mit dem Office im Name angedeutet, wurde vor allem der Fokus wieder auf die Dokumentenmanagement-Komponente gelegt. Es fanden aber auch im Bereich des Enterprise 2.0-Funktionalitäten starke Überarbeitungen statt.

So wurden bei dieser Version wichtige Social Software-Anwendungen, wie Wiki und Blog hinzugefügt. Die neuen Anwendungen zeichneten sich aber noch durch viele Mängel aus, die durch unzählige extern entwickelte Lösungen verbessert wer-

den mussten. Wie auf Codeplex[23] zu sehen sind zum MOSS unzählige Verbesserungen zu diesem Thema erstellt wurden.

Technologisch basierte der MOSS wieder auf der Grundlage der Windows SharePoint Services, die als WSS v2 in einer neuen Version erschien.

Produktbeschreibung

Bei der aktuellen Version des SharePoint Servers bildet weiterhin eine überarbeitete Ausführung der Windows SharePoint Services die technologische Grundlage der Anwendung. Die neue Ausgabe der WSS wurde als SharePoint Foundation 2010 veröffentlicht und steht zum kostenlosen Herunterladen bei Microsoft[24] zur Verfügung.

Das Leistungsvermögen bzw. Anwendungsspektrum des SharePoint Server 2010 fasst Microsoft in folgende sechs Bereiche zusammen:

„Sites: *SharePoint Sites bietet eine einzelne Infrastruktur zur Bereitstellung von Portal- und Zusammenarbeitsfunktionen in Intranets, Extranets sowie auf Internetseiten. Benutzer können hier unternehmensübergreifend Informationen, Daten und Fachkenntnisse miteinander austauschen.*

Composites: *SharePoint Composites ermöglicht es Benutzern, umgehend auf Unternehmensanforderungen zu reagieren, indem es mithilfe einer umfangreichen Auswahl an Bausteinen, Tools und Self-Service-Funktionen vor Ort oder in der Cloud eigene Lösungen erstellt, SharePoint 2010 ohne Programmier-Aufwand an Ihre Anforderungen anzupassen.*

Insights: *SharePoint Insights ermöglicht Benutzern den Zugriff auf und die Interaktion mit Informationen über unstrukturierte und strukturierte Datenquellen hinweg.*

[23] http://www.codeplex.com, verfügbar am 10.05.2010 – Hosting-Webseite von Microsoft auf der quelloffene Programme veröffentlicht werden können.
[24] http://technet.microsoft.com/de-de/sharepoint/ee263910.aspx, verfügbar am 15.06.2010

Sie unterstützen Benutzer dabei, die richtigen Personen mit dem richtigen Fachwissen zu finden, damit sie Geschäftsentscheidungen besser und flexibler treffen können.

Communities: *SharePoint Communities ermöglicht Benutzern noch effektiver zusammenzuarbeiten. Damit können sie ganz einfach Gruppen bilden, Wissen und Ideen austauschen, Kontakt mit Kollegen aufnehmen und Informationen sowie Experten mühelos finden.*

Content: *SharePoint Content ermöglicht allen Benutzern die Teilnahme an einem verwalteten und kompatiblen Content Management-Lebenszyklus. Mit SharePoint Content kann die Verwendungsweise optimal mit Richtlinien und Prozessen in Einklang gebracht werden.*

Search: *SharePoint Search ermöglicht es Benutzern – dank einer integrierten und leicht verwaltbaren Plattform – die gewünschten Inhalte, Informationen und Personen zu finden. Mit einer neuen erstklassigen Technologie für die Suche im Unternehmen wird das möglich.“* (Microsoft 2010b)

Abbildung 4-1: Leistungsvermögen/Anwendungsspektrum SharePoint Server 2010 (Microsoft 2010b)

Zusätzlich kann beim Microsoft SharePoint Server 2010 zwischen *Standard Edition* und *Enterprise Edition* gewählt werden. Wie die Namen schon verraten, handelt es

sich bei der Standard Edition um die Standardausführung, mit allen bereits aufgeführten Funktionen, und bei der Enterprise Edition um eine mit besonderen Unternehmensfunktionalitäten erweiterte Version.

4.1.1.2 Betrachtung der Social Software-Funktionalitäten

Für die Untersuchung der Funktionalitäten wurde zunächst eine Testumgebung eingerichtet. Dabei wurde eine sogenannte virtuelle Maschine erstellt, in der ein Microsoft Windows Server 2008 R2 64-Bit-Version mit Microsoft SharePoint Server 2010 Enterprise Edition installiert wurde. Zur Virtualisierung der Maschine wurde die kostenlose Software VMware Player genutzt.

Vorteil bei dieser Konstellation ist es, dass für die Evaluierung kein extra Rechner benötigt wird. Stattdessen ermöglicht eine Virutalisierungs-Software, dass parallel zur Laufzeit zu dem lokalen Betriebssystem auf dem Rechner ein zweites Betriebssystem betrieben werden kann.

Bei der ersten oberflächlichen Betrachtung wurde schnell klar, dass Microsoft die im MOSS bereits eingeführten Social Software-Funktionalitäten konsequent weiterentwickelt bzw. neue Funktionalitäten hinzugefügt hat.

Eine SharePoint-Seite sieht wie in Abbildung 4-2 aus. Weitere detaillierte Bilder zu den untersuchten Funktionalitäten befinden sich im Anhang A – Bilder Microsoft SharePoint Server 2010.

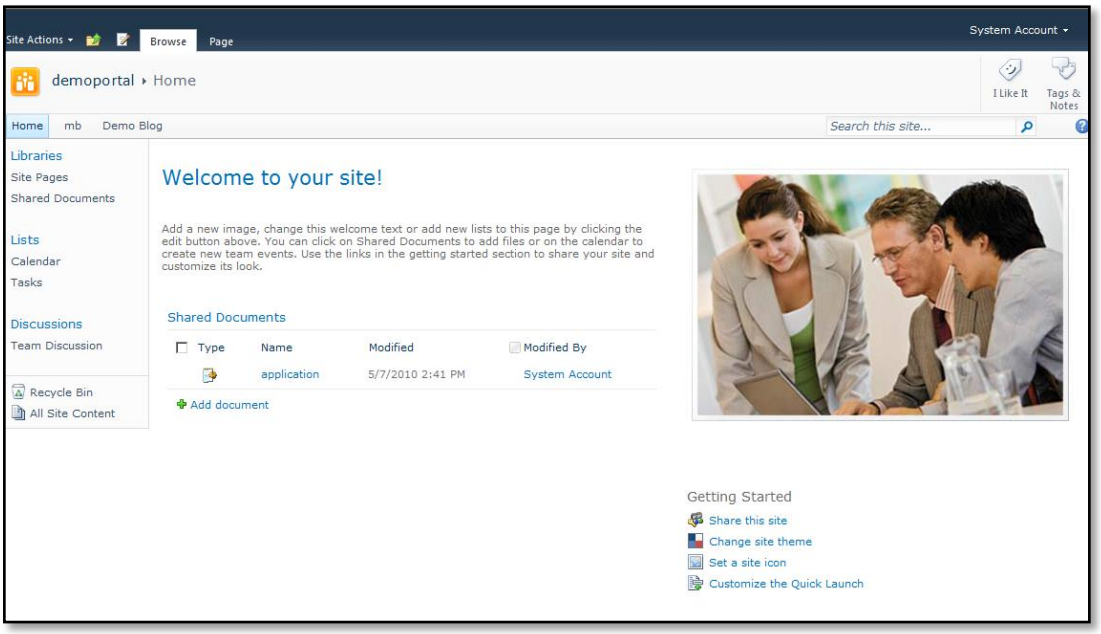


Abbildung 4-2: SharePoint Demo-Website

Technologie

Da zwischen den Veröffentlichungen neuer SharePoint Server-Versionen in der Regel mehrere Jahre liegen, unterliegen die technologischen Veränderungen ebenfalls dem gleichen Zeitraum. Wobei die 2010er Version erst vor kurzen erschienen ist und die derzeit aktuellen Technologien unterstützt werden. Microsoft setzt bei seinen wichtigen Produkten meist auf etablierte Techniken, dies ist auch beim SharePoint der Fall. So sind zwar kaum experimentelle Technologien zu finden, dafür werden die Aktuellen konsequent eingesetzt.

Der SharePoint Server unterstützt auf Client-Seite die aktuellen Versionen der Web-Browser Mozilla Firefox, Microsoft Internet Explorer und eingeschränkt Apple Safari.

Ajax: Ajax wird überall eingesetzt, wo man es erwartet. Es ist kein unnötiges Neu-Laden feststellbar.

RSS/Atom: Nicht nur bei den Social Software-Anwendungen, sondern auch an anderen Stellen, wie bei Listen und Kalendern wird eine Funktion zum Abonnieren der Inhalte angeboten. Als Technologie wird der RSS 2.0 Standard verwendet. Zusätzlich können an vielen Stellen sogenannte Alerts (dt. Alarmsignale) eingestellt werden, bei denen nach einer Auswahl von Bedingungen E-Mail oder SMS an einen Empfänger bei Eintreffen der Ereignisse geschickt wird.

Weitere anwendungsübergreifende Anforderungen

Der SharePoint Server 2010 macht bei anwendungsübergreifenden Funktionen in allen Belangen einen sehr reifen Eindruck.

Kollaborationsbereiche: Die logische Struktur der Webkomponente des SharePoint Server lautet wie folgt. Innerhalb eines Server können mehrere Webanwendungen verwaltet werden, die wiederum aus Websitesammlungen (Site Collections) bestehen (siehe Abbildung 4-3). Diese stellen eine Menge von logisch zusammenhängenden Websites, den Kollaborationsbereichen, dar. Nutzer mit den entsprechenden Rechten können einfach neue Websites erstellen und so auch leicht Bereiche erweitern. Dabei ist es möglich aus zahlreichen Vorlagen zu wählen. Die Kollaborationsbereiche können hierarisch angeordnet werden und es stehen sehr umfangreiche Funktionen zu Verwaltung der Websites bereit. So besitzen die Website-Administratoren die Option, eigene Rechtegruppen anzulegen oder bereits bestehende Gruppen zu bearbeiten.

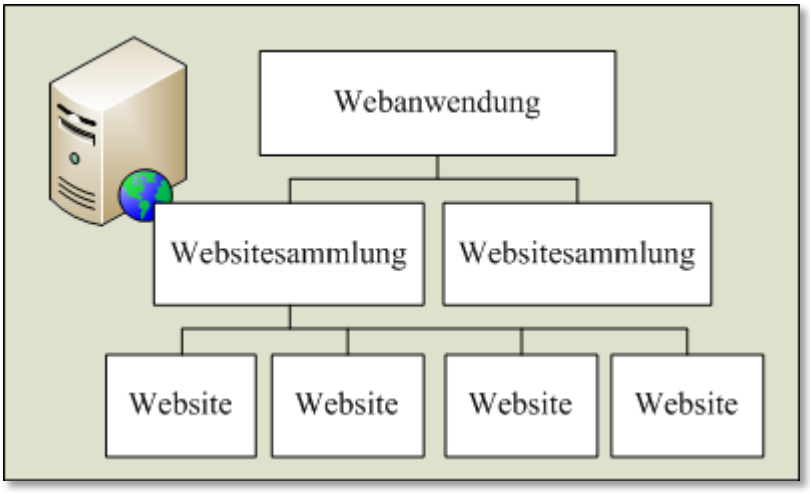


Abbildung 4-3: Struktur einer SharePoint Webanwendung

WYSIWYG-Editoren: Beim Bearbeiten von Texten kommen durchgehend bei allen Social Software-Anwendungen hochwertige WYSIWYG-Editoren zum Einsatz. Die Editoren sind von der Menü-Führung an die neueste Version von Microsoft Office Word angelehnt. Außerdem können die meisten Texte direkt in Word bearbeitet werden. Die Anforderungen an die Editoren werden mehr als erfordert erfüllt.

Mehrsprachigkeit: Es können speziell für Websites regionale Einstellungen getätigt werden. Zusätzlich steht es den Nutzern frei, zwischen vorinstallierten Anzeigesprachen zu wählen. Die Mehrsprachigkeit wird im Sinne der Anforderungen voll erfüllt.

Globale Suchfunktion: Eine globale, anwendungsübergreifende Suchfunktion steht zur Verfügung. Die zu dursuchenden Bereiche können gewählt und auch die geforderten Filter eingestellt werden. Zusätzlich besteht die Möglichkeit Suchergebnisse zu abonnieren und Bereiche zum Durchsuchen mit dem Windows Explorer zu synchronisieren.

Anwendungsspezifische Anforderungen

Die Social Software-Anwendungen können über die Website-Vorlagen erstellt werden oder sind anderweitig in die Plattform eingebunden.

Blogs: Die Bloganwendungen werden in den Bereichen über Seitenvorlagen eingerichtet. Die Einträge erfüllen die Anforderungen, lediglich eine Bewertung ist ohne Anpassungen nicht möglich. Die Kommentar-Funktion ist auch eher schlicht gehalten. Es werden keine Antworten auf Kommentare visualisiert und Bewertungen sind nicht möglich.

Eine Blogroll kann angelegt werden und mehrere Linksammlungen können parallel verwaltet werden. Trackbacks bzw. Pingbacks werden ohne Plug-ins nicht unterstützt. Permalinks gehören genauso, wie eine Archivfunktion, Kategorien, Tags und Abonnement-Funktion zum Repertoire des SharePoint-Blogs. Allerdings ist das Abonnieren von Kommentaren nur umständlich möglich.

Für die Suche kann auf die globale Suche mit ausgewählter Blog-Seite zurückgegriffen oder ein extra Webpart[25] dafür genutzt werden. Über die umfangreiche Rechtevergabe der SharePoint-Seiten bzw. Erstellung von Rechtegruppen können für die Blog-Website die entsprechenden Rollen erstellt werden. Zusätzlich besteht die Option, dass die Nutzer über die eigene Profilseite einen Blog erstellen.

Mircoblogs: Eine reine Microblog-Anwendung für die Nutzer steht nicht zur Verfügung. Es besteht nur die Möglichkeit Statusmeldungen in den Profilseiten der Nutzer zu speichern. Diese sind dann für alle sichtbar. Mit Hilfe der Kollegenliste und einem integrierten Newsfeed, können die vernetzten Nutzer über Statusmeldungen informiert werden.

Wikis: Wenn eine Wiki-Bibliothek innerhalb eines Kollaborationsbereichs eingerichtet wird, dürfen dort auch Wiki-Seiten von den Nutzern erstellt und verlinkt

[25] Mit sogenannten Webparts können Inhalte bzw. Anwendungen in den Webseiten von den Nutzern eingeblendet bzw. ausgeblendet werden – dazu mehr in Kapitel 5.1.2.2, S. 82.

werden. Zusätzlich kann eigens ein Enterprise-Wiki von den Administratoren angelegt werden. Beim Erstellen der Artikel werden alle Anforderungen erfüllt.

Auch das Anhängen von Dateien ist realisierbar. Außerdem können mit Hilfe der Webpart-Technologie allmögliche statische und dynamische Inhalte, wie z.B. Silverlight Videos oder RSS-Feed, mitten im Text eingefügt und angezeigt werden. Für die Qualitätssicherung stehen Funktionen zur Versionskontrolle und zur Kommentierung bereit. Diskussionen sind zu einem Artikel sind nur sehr umständlich möglich. Herausgehoben werden sollte hier eine Funktion mit der die aktuelle Version eines Artikels mit einer beliebigen älteren Version auf Unterschiede verglichen werden kann. Der Reifegrad wird beim Enterprise-Wiki anhand der Bewertung der Nutzer abgelesen.

Mit Hilfe von Webparts sind so ziemlich alle möglichen Kombinationen für eine Übersichtsseite realisierbar. Leider stehen keine vorkonfigurierte Vorlagen für eine Übersichtsseite bereit.

Die Verlinkung ist funktionell. Es steht eine einfache Wiki-Syntax bereit, bei der bei der Eingabe bereits existierende Wiki-Seiten vorgeschlagen werden. Ferner werden Namensveränderungen automatisch erkannt. Links zu Wiki-Artikeln in anderen Bereichen funktionieren nicht unter Verwendung der Wiki-Syntax. Darüber hinaus können Links zu einem Artikel angezeigt werden.

SharePoint ermöglicht jedem Wiki-Artikel Tags zu zuweisen, Kategorien nur im Enterprise-Wiki. Es können Alerts für Veränderungen an den Artikeln erstellt und ein Wiki kann per RSS abonniert werden. Das Exportieren von Wiki-Inhalten ist nur mit Installieren von Plug-ins möglich.

Als Suche kann die globale Suche genutzt werden. Es werden dann entsprechend die gewünschten Wikis ausgewählt. Beim Rollenmanagement können analog zu Blogs die verschiedenen Rollen durch selbsterstellte Rechtegruppen erzeugt werden. Es bestehen keine besonderen Funktionen, die eine extra Wiki-Gärtner-Rechte-Gruppe benötigen. Außerdem können so viele Wikis wie gewünscht erstellt werden.

Social Networking-Dienste: Die Social Networking-Dienste und allem voran das Nutzerprofil ist eine Stärke des aktuellen SharePoint Server. Es können externe Profile und Profildaten importiert werden. Aber auch die Nutzer selbst können ein Profil anlegen. Den Administratoren stehen verschiedene Funktionen zur Verfügung, mit denen unterschiedliche Profiltypen verwaltet und Profilfelder definiert werden können. Auch Einstellungen zur Privatsphäre der Profilinformationen sind möglich.

Zusätzlich gibt es eine Anzeige für die hierarische Position der Nutzers innerhalb der Organisation. Darüber werden die Verbindungen zwischen den Nutzern veranschaulicht. Daneben können diese eine eigene Kollegenliste erstellen, für die aber keine grafische Anzeige angeboten wird. Die selbst erstellten Verbindungen untereinander werden z.B. bei der Suche nach anderen Personen genutzt, wobei dort immer die Nutzer mit der geringsten sozialen Distanz[26] zuerst angezeigt werden. Ebenfalls gibt es eine Anzeige auf der Profilseite, die gemeinsame Kontakte mit der jeweiligen Person auflistet.

Auch Activity Streams werden unterstützt, hier unter dem Namen Activity Feed bzw. Newsfeed. Zudem können umfangreiche Einstellungen zu den Aktivitäten getätigt werden. Auf der Profilseite werden die letzten Aktivitäten der Nutzer angezeigt.

Die Nutzer können Nachrichten auf den Profilseiten hinterlassen. Ein eigenständiges System für private Nachrichten ist nicht in den SharePoint Server eingebunden. Im Gegensatz dazu ist eine konsistente Identitätsverfolgung möglich, denn alle Aktivitäten in den einzelnen Social Software-Anwendungen sind direkt mit dem Profil des jeweiligen Nutzers verbunden. Badges werden nicht unterstützt. Ebenfalls können den Profilen keine Tags zugewiesen werden.

Social Tagging: Es können Seiten, wie bei allen anderen Inhalten, die im SharePoint verwaltet werden, mit Tags markiert werden. Dabei ist auch die Unterschei-

[26] Der Abstand zu einem anderen Nutzer, wenn die eigenen Verbindungen zu anderen Nutzern, deren Verbindungen zu weiteren Usern und so weiter einbezogen werden.

dung zwischen privat und öffentlich einstellbar. Die Schlagworte werden aus bereits vergebenen oder extra gespeicherten Schlagworten vorgeschlagen. Über die gleiche Funktion lassen sich Notizen hinzugefügen.

Über Webparts können Tag Clouds an verschiedensten Stellen eingebunden werden. Ebenfalls werden auf den Profilseiten der Nutzer deren vergebene Tags visualisiert. Im Newsfeed sind Tags abonnierbar. Für die einzelnen Schlagworte kann jeweils eine eigenständige Seite angezeigt werden.

Der eingebundene Social Bookmarking-Dienst ist mit der Tagging-Anwendung verschmolzen. So können auch externne Seiten Tags und Notizen zugewiesen werden. Dafür muss eine extra Link im Browser unter den Favoriten gespeichert werden.

Weitere Anwendungen: Zu den zusätzlichen Social Software-Anwendungen, die im SharePoint Server angeboten werden, gehören Foren. Diese sind simple Bulletin Boards und können in den Kollaborationsbereichen einfach angelegt werden. Umfragen sind als eigenständige Seiten realisiert.

Eine Instant Messaging-Software wird nicht angeboten. Collaborative Filtering wird nicht merkbar eingesetzt.

Um simultan SharePoint-Inhalte bearbeiten zu können, also als vollwertiger Gruppeneditor, ist Microsoft Word nutzbar. Dabei handelt es sich jedoch um ein eigenständiges Programm, was unabhängig vom SharePoint Server für die Clients erworben werden muss.

4.1.2 IBM Lotus Connections

Lotus Connections ist ein Produkt der Firma IBM und wird als „Social Software für Unternehmen“ (IBM 2010) vermarktet. Ebenso wie der SharePoint für Microsoft eine Erfolgsgeschichte darstellt, ist auch Connections eins der erfolgreichsten Soft-

ware-Produkte von IBM. Dies unterstreicht ein Rekordzuwachs der Software nach Veröffentlichung. (vgl. Boulton 2008) Die aktuelle Version 2.5 erschien im August 2009.

4.1.2.1 Allgemeines zu Lotus Connections

Bei Lotus Connection handelt es sich um ein recht junges Produkt. Die erste Version 1.0 wurde erst im Sommer 2007 veröffentlicht und im Gegensatz zum SharePoint wurde Connections von Anfang an als Social Software-Kollaborationsplattform geplant. Inzwischen sind zwei weitere wichtige Versionen erschienen, Version 2.0 und 2.5, in denen zusätzliche Anwendungen und Verbesserungen hinzugefügt wurden.

Doch auch bei den neuen Versionen beschränkt sich das Anwendungsspektrum auf Social Software, denn IBM verfolgt einen anderen Ansatz als Microsoft bei ihrem SharePoint. Es wird nicht eine Plattform, die so viele wie mögliche Enterprise-Funktionalitäten erfüllt, als einzelnes Produkt angestrebt. Vielmehr stehen mit Anwendungen wie Lotus Quickr und Lotus Sametime weitere Produkte bereit, die zusammen über diverse Schnittstellen zu einer multifunktionalen Plattform verbunden werden können.

4.1.2.2 Betrachtung der Social Software-Funktionalitäten

Zur Evaluierung der Funktionalitäten von Lotus Connection konnte keine lokale Installation eingerichtet werden, stattdessen wurde mit Lotus Greenhouse[27] auf eine online verfügbare Testumgebung zurückgegriffen. Diese wird direkt von IBM zum Testen angeboten und setzt lediglich eine Registrierung voraus.

[27] https://greenhouse.lotus.com/, verfügbar am 10.06.2010

Eine Lotus Greenhouse-Seite sieht wie in Abbildung 4-4 aus. Weitere detaillierte Bilder zu den untersuchten Funktionalitäten befinden sich im Anhang B – Bilder IBM Lotus Connections.

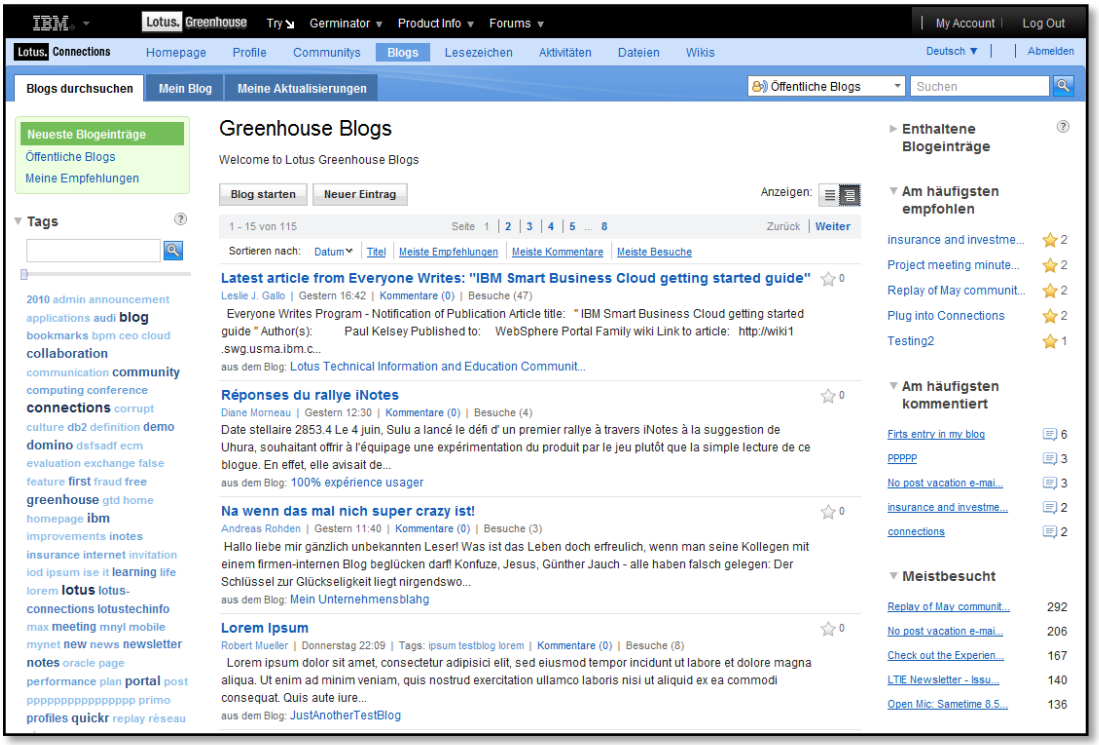


Abbildung 4-4: Lotus Greenhouse – Lotus Connections

Technologie

Lotus Connection macht einen reifen Eindruck und ist technologisch auf dem aktuellen Stand. So werden die weit verbreiteten Web-Browser, Microsoft Internet Explorer, Mozilla Firefox und Apple Safari, in der neusten Version voll unterstützt.

Ajax: An den üblichen Stellen, wie beim vorschlagen von Tags wird Ajax verwendet. Es gibt kein sinnloses Neu-Laden von Seiten.

RSS/Atom: Feeds können an unterschiedlichen Stellen abonniert werden. Dabei wird auf das Atom-Format gesetzt.

Weitere anwendungsübergreifende Anforderungen

Die Anwendungen sind zwar übergreifend miteinander verbunden, jedoch sind die Inhalte je Anwendungstyp einzeln aufgelistet.

Kollaborationsbereiche: Sogenannte Communities bilden die Kollaborationsbereiche. In diesen können die Social Software-Anwendungen eingebunden werden und umfangreiche Rechte-Management-Einstellungen getätigt werden. Die Communities sind nicht hierarisch angordbar.

WYSIWYG-Editoren: Die Editoren unterscheiden sich zum Teil. So erfüllt der Wiki-Editor die Anforderungen mehr als erforderlich, der Blog-Editor hat jedoch weniger Funktionen. Es fehlen zum Beispiel Format-Vorlagen. Im Gesamtpaket machen die Editoren allerdings einen guten Eindruck, denn Optionen wie Datei anhängen, Bilder einfügen, etc. sind überall möglich.

Mehrsprachigkeit: Die Nutzer können selbst Einstellungen zur Menü-Sprache tätigen, aber mehrsprachige Blogs oder Wikis werden von Haus aus nicht unterstützt.
Globale Suchfunktion: Eine anwendungsübergreifende Suche, welche neben dem vorausgesetzten Filter weitere Einschränkungen wie nach personenbezogenen Inhalt und Tags ermöglicht, wird angeboten.

Anwendungsspezifische Anforderungen

Ableger der Anwendungstypen können entweder in den Communities oder unabhängig davon eingerichtet werden.

Blogs: Die Blogeinträge haben die typische Form und dazu eine extra Anzeige für Aufrufzahlen. Außerdem können die Nutzer einen Eintrag empfehlen, was mit Hilfe von Sternen visualisiert wird. Die Kommentare sind übersichtlich, aber eine Dialog-

funktion wurde nicht eingearbeitet. Dafür ist aber das Empfehlen der Kommentare, analog wie bei den Blog-Einträgen möglich.

Die Erstellung einer Blogroll ist möglich, aber nicht die Verwaltung mehrerer solcher. Eine nützliche Option ist das automatische Vorschlagen von ähnlichen Blogs. Permalinks werden unterstützt, ferner stehen diese sogar für die Kommentare zur Verfügung. Trackpacks bzw. Pingbacks sind nicht in das System eingebunden.

Ein Archiv wird automatisch erstellt und passt sich dem Blog an. Kategorien für die Blog-Einträge können nicht geführt werden, jedoch ist das Zuweisen von Tags realisierbar.

Beim Abonnieren werden gleich drei unterschiedliche Optionen angeboten: Der Nutzer kann entweder einen Feed für alle Blogeinträge eines Blogs abonnieren, oder die Kommentare nur eines Blogs auswählen oder die Kommentare eines bestimmten Blogeintrags bestellen.

Für jeden Blog besteht eine eigenständige Suche, welche auch auf einen Kommentar beschränkbar ist. Ebenfalls können die eigenen und alle öffentlichen Blogs der Anwendung durchsucht werden.
Es stehen die drei Standardrollen Autor, Editor, Leser bereit, welche über die Communities vergeben bzw. verwaltet werden. Je Kollaborationsbereich kann ein Blog erstellt werden, losgelöst von einem Bereich ist es möglich mehrere pro Nutzer zu erstellen.

Mircoblogs: Es sind keine Microblogs innerhalb der Kollaborationsbereiche möglich. Nur über das Profil sind Statusmeldungen der Nutzer mit ihrem Netzwerk austauschbar. Dialoge sind innerhalb dieses Systems möglich und werden visuell sogar hervorgehoben. Jedoch beschränkt sich die Microblog-Anwendung auf diese Funktion.

Wikis: Bei der Erstellung eines Wiki-Artikels stehen keine Gliederungsvorlagen, sondern nur Formatvorlagen zur Auswahl. Es können Multimedia-Inhalte über Adobe Flash eingebunden und einfach Dateien an einen Artikel angehangen werden

Zur Qualitätssicherung wird eine gute Versionskontrolle, bei der der direkte Vergleich älterer Versionen möglich ist, aufgeboten. Unterstützt wird dies dadurch, dass die Nutzer den Artikeln einfach Kommentare hinzufügen können

Eine direkte Übersichtsseite ist nicht möglich, stattdessen werden in den Communities zum Beispiel die zuletzt bearbeiteten Seiten aufgelistet. Kategorien und Tags sind den Wiki-Artikeln zuordbar. Eine hierarische Anordnung der Artikel ist möglich.

Die Artikel lassen sich über eine einfache Wiki-Syntax miteinander verlinken, dies ist aber nur innerhalb eines Wikis möglich. Beim Abonnieren kann ein Artikel, sowie dessen Kommentare und auch eine komplettes Wiki gewählt werden.

Bei der Suche kann ein bestimmtes Wiki, alle eigenen oder alle öffentlichen Wikis durchsucht werden. Das Rollen-Management ist dem der Bloganwendung gleich und wird über die Communities angeboten, somit ist also eine separate Rollenvergabe für Blog und Wiki innerhalb eines Kollaborationsbereiches nicht möglich. Genauso kann ein Wiki pro Community, aber losgelöst davon mehrere erstellt werden.

Social Networking-Dienste: Für jeden Nutzer wird automatisch ein Profil erstellt. Einstellungen zur Privatsphäre, also welche Daten für wen sichtbar sind, sind nicht vorhanden. Eine interessante Idee ist die Aussprache-Funktion. Bei dieser kann eine Audiodatei mit dem eigenen Namen ans Profil angefügt werden. Ebenfalls nützlich ist, dass Profile als sogenannte vCard herunterladbar und anschließend in Programmen wie Outlook importierbar sind.

Mit der Berichtskette kann eine vorgefertigte organisatorische Vernetzung angezeigt werden, diese beschränkt sich jedoch auf die Vorgesetzten. Außerdem kann

auch eine Kollegenliste erstellt werden. Für die Visualisierung der Liste ist die zusätzliche Anwendung IBM Atlas notwendig, welche eine sehr umfangreiche Darstellung eines Social Graph ermöglicht.

Auf den Profilseiten der Nutzer werden die letzten Aktivitäten der Nutzer angezeigt. Zusätzlich ist es über die Startseite möglich, nach Personen und Tags zu filtern. Es ist nicht möglich, auf das bereits vorhandene Netzwerk zurückzugreifen.

Zur Kommunikation zwischen den Nutzern können Notizen auf den Profilseiten geschrieben werden. Ein privates Nachrichtensystem gehört nicht zur Ausstattung von Lotus Connections.

Eine konsistente Identitätsverfolgung wird im Sinne der Anforderungen vollkommen erfüllt, sogar die Badges werden angezeigt. Auch Tags können einem Nutzerprofil hinzugefügt werden, wobei der Besitzer des Profils wie erfordert die Kontrolle über diese besitzt.

Social Tagging: Es kann so gut wie bei jedem Inhalt bei der Erstellung Tags zugewiesen werden, wobei auch ein Vorschlagesystem eingesetzt wird. Bei der Anzeige kann dann zwischen Listenform, mit nummerischer Häufigkeitsanzeige, oder Tag-Cloud gewählt werden. Jedoch sind nicht jedem bereits existierenden Inhalt neue Tags zuweisbar und auch private Tags sind nicht möglich.

Für das Social Bookmarking wird eine eigenständige Anwendung innerhalb der Plattform angeboten, die genauso in die Communities integriert werden kann. So können mehrere Linksammlungen geführt werden.

Weitere Anwendungen: Instant Messaging gehört nicht zur Ausstattung von Lotus Connections. Jedoch kann die Instant Messaging-Software Lotus Sametime über die Profilseiten in Connections integriert werden. Gruppeneditoren gehören genauso, wie Collaborative Filter, nicht zum Repertoire von Lotus Connections.

Diskussionsforen sind ein fester Bestandteil der Plattform. Dabei handelt es sich um Bulletin Boards, die zusätzlich Dialoge innerhalb eines Threads hervorheben. Bewertungsfunktionen oder ähnliches werden nicht ermöglicht.

4.1.3 Atlassian Confluence

Confluence ist ein Produkt der Firma Atlassian. Primär als Enterprise-Wiki-Software angepriesen, aber auch mit anderen Social Software-Funktionalitäten ausgestattet, zählt die Anwendung mittlerweile bei Experten zum führenden Kreis der Kollaborationsplattformen. (vgl. Forrester 2009) Seit Mai 2010 kann das Produkt als Version 3.2.1 erworben werden.

4.1.3.1 Allgemeines zu Confluence

Die erste Ausführung von Confluence, die 2004 erschienen ist, war eine reine Wiki-Anwendung für Unternehmen. Atlassian gehört seitdem zu den Marktführern in diesem Segment. Weitere Social Software-Dienste und Anwendungen wurden erst mit späteren Aktualisierungen hinzugefügt.

Das Programm wurde in Java geschrieben und von Anfang an wurde ein Schwerpunkt darauf gelegt, dass eine einfache Erweiterung der Software möglich ist. So gehören Plug-ins, mit denen einfach unter anderen Kollaborations-, Dokumentations- und Social Software-Funktionalitäten hinzugefügt werden können, zur besonderen Stärke des Produktes.

4.1.3.2 Betrachtung der Social Software-Funktionalitäten

Die Evaluierung von Confluence wurde anhand einer von Atlassain online gehosteten Sandbox-Lösung getätigt.[28] Dabei handelt es sich um eine Seite, die jeder Internet-Nutzer verwenden kann, wobei die Software für jeden abgeschirmt vom Restsystem bzw. versionsgesichert läuft. Ein Teil der Funktionalitäten setzt eine vorherige Anmeldung voraus.

Eine Atlassain Confluence-Seite in der Sandbox sieht wie in Abbildung 4-5 aus. Weitere detaillierte Bilder zu den untersuchten Funktionalitäten befinden sich imAnhang C – Bilder Atlassian Confluence.

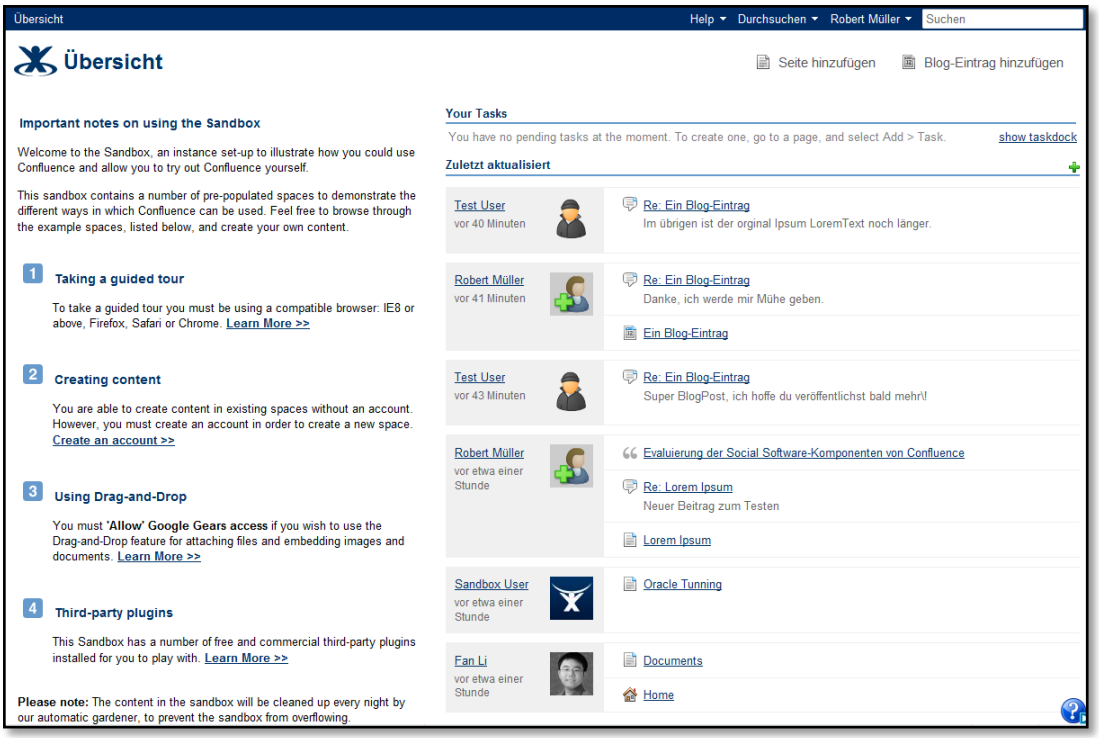


Abbildung 4-5: Atlassain Confluence – Sandbox

[28] http://www.atlassian.com/sandbox/confluence-tracker.jsp, verfügbar 15.05.2010

Technologie

Wie der SharePoint Server 2010 und Lotus Connections unterstützt Confluence genauso die aktuellsten Versionen der Web-Browser Microsoft Internet Explorer, Mozilla Firefox und Apple Safari.

Ajax: Der Einsatz von Ajax findet an allen wichtigen Stellen statt. Zusätzlich wird mit Gears eine Technologie von Google unterstützt, mit der nach Installation eines Browser Plug-Ins Dateien per Drag & Drop vom Desktop direkt in eine Seite kopiert werden können.

RSS/Atom: Feed-Benachrichtigungen werden über die Kollaborationsbereiche ausgewählt. Zusätzlich ist es möglich mit einem Feed-Builder eigene Feeds zu erstellen. Das verwendete Feed-Format ist Atom.

Weitere anwendungsübergreifende Anforderungen

Die Basis von Confluence ist eine Wiki-Engine, dies zieht sich durch die gesamte Anwendung.

Kollaborationsbereiche: Die Kollaborationsbereiche werden in Confluence als Spaces bezeichnet. Es können verschiedene Einstellungen zu den Zugriffsrechten getätigt werden. Hierarisches Anordnen der Spaces ist nur unter Benutzung von Plug-Ins möglich.

WYSIWYG-Editoren: Dort wo ein Editor zu Textbearbeitung notwendig ist, wird der Inhalt meist in Wiki-Form gespeichert. So kann überall der Wiki-WYSIWYG-Editor verwendet werden – auch bei Blog-Einträgen. Mit Hilfe eines von Atlassain bereit gestellten Browser-Plug-in können die Inhalte auch per Microsoft Word bearbeitet werden. Der originale Editor ist eher durchschnittlich. Die wichtigsten Funktionen beinhaltet er aber und kann mit Hilfe von Plug-ins verbessert werden.

Mehrsprachigkeit: Die Nutzer können selber die Menü-Sprache auswählen. Das Erstellen und Verwalten von mehrsprachigen Inhalten ist nur mit Hilfe von Plug-Ins möglich.

Globale Suchfunktion: Es wird eine anwendungsübergreifende Suche angeboten, die neben dem vorausgesetzten Filter ein Einschränkung nach personenbezogenen Inhalt ermöglicht.

Anwendungsspezifische Anforderungen

Über ein Kontext-Menü können Wiki-Seiten, Blog-Einträge sowie Kommentare einfach eingefügt werden.

Blogs: Die Blog-Einträge werden auf Grundlage von Wiki-Seiten realisiert, so stehen umfangreiche Mittel zur Gestaltung der Posts zur Verfügung. In die Artikel lädt man vorgefertigte Programme, sogenannte Makros. Mit diesen kann ein breites Spektrum an Informationen, zum Beispiel einer Inhaltsliste, dargestellt werden.

Die Bewertung von Einträgen und Kommentaren ist ohne Plug-in nicht möglich. Dialoge werden innerhalb der Kommentare in besonderer Form dargestellt.

Trackback bzw. Pingback werden nicht unterstützt, dafür aber Permalinks und Blogrolls. Die Linksammlung basiert dabei auf den mit Hilfe der Social Bookmarking gesammelten Links, kann aber durch Angabe von Schlagworten eingeschränkt werden.

Ein Archiv, durch das unter Verwendung eines Kalenders navigiert werden kann, wird automatisch erzeugt.

Tags und Tag-Cloud sind möglich, aber die Einteilung von Kategorien wird daneben nicht umgesetzt. Beim Abonnieren kann zwischen Blog-Einträgen und Kom-

mentaren ausgewählt werden. Suche, Rollen-Management und Verwaltung erfüllen die erforderten Anforderungen.

Mircoblogs: Es kann nur ein vom Benutzer bezogener Status aktualisiert werden, der über den Activity Stream und dementsprechend über den abonnieren von Kontakten geteilt werden. Bei der Suche stehen auch die Statusaktualisierungen zur Verfügung.

Wikis: Dank der Makros, die überall in die Artikel geladen werden können, lässt sich eine Übersichtseite leicht erstellen. Die Möglichkeit zur Erstellung und Verwaltung von Vorlagen besteht. Alle Wiki-Inhalte können kommentiert, sowie in Dialog-Form diskutiert werden. Ebenfalls ist eine Versionskontrolle mit Vergleich von älteren Versionen möglich.

Den Wiki-Artikeln können Tags zugewiesen werden. Mit einem entsprechenden Makro werden diese auch als navigierbare Kategorien verwendet. Der Feed-Builder ermöglicht neben einen einzelnen Wiki, nach Wunsch ausgewählte Wikis zu abonnieren.

Die Option Wiki-Seiten als PDF und Word-Dokument zu exportieren besteht. Weiterhin ist auch die Auswahl mehrerer Seiten realisierbar. Bei dieser Variante muss allerdings aus den Formaten PDF, HTML und XML gewählt werden.

Beim Durchsuchen wird zwischen allen Wiki-Inhalten oder allen eines Kollaborationsbereichs unterschieden. Die Rollen können anhand von Benutzergruppen und der umfangreichen Rechtevergabe verteilt werden.

Die ganze Plattform ist quasi ein Wiki. So ist es möglich, dass auch einfach Wiki-Seiten Kollaborationsbereich übergreifend verlinkt werden und an jeder Stelle eine Wiki-Seite angelegt wird.

Social Networking-Dienste: Es wird für jeden Nutzer automatisch eine Profil-Seite angelegt. Dort können verschiedene Daten eingegeben werden. Einstellungen zur Sichtbarkeit der Felder sind nicht möglich.

Die Vernetzung ist eher schwach ausgeprägt, die Kollegenliste ist auf die Funktionen Folgen und Nicht-Folgen beschränkt. Eine Visualisierung des Netzwerks ist nicht vorgesehen.

Der Activity Stream wird über das Profil realisiert. Dort ist allerdings nicht einstellbar, welche Aktivitäten angezeit werden. Der Stream darf per Feed abonniert werden. Außerdem können angepasste Activity Streams in die Wiki-Seiten eingebunden werden, die auf einen Kollaborationsbereich und bzw. oder auf eine bestimmte Liste von Nutzern beschränkt werden können.

Es besteht keine Option Notizen oder direkte Nachrichten zwischen den Nutzern auszutauschen. Eine konsistente Identitätsverfolgung im Sinne der Anforderung erfolgt und auch Badges werden angezeigt. Die Zuweisung von Tags auf ein Nutzerprofil ist nicht möglich.

Social Tagging: Allen Wiki-Seiten, die die Basis für den Webseiten-Inhalt darstellen, können Notizen und Tags zugewiesen werden. Dabei werden eine Liste von bereits vorhandenen Tags und weitere Vorschläge bei der Eingabe angezeigt. Private Tags sind nicht möglich. Die Tags dürfen entweder als Tag-Cloud oder als Liste dargestellt werden.

Die integrierte Social Bookmarking-Anwendung ist direkt mit den Spaces verbunden, wodurch auch mehrere Linksammlungen möglich sind.

Weitere Anwendungen: Instant Messaging, Gruppeneditoren und Collaborative Filtering gehören nicht zum Repertoire der Anwendung. Foren sind nur mit Hilfe von Plug-Ins in Confluence realisierbar.

4.2 Die Bewertungsmatrix

Nachdem nun alle Produkte vorgestellt wurden und ein Einblick in die Umsetzung der Social Software-Funktionalitäten gegeben wurde, sollen die Stärken und Schwächen der Produkte im Vergleich zueinander in einer visualisierten Form veranschaulicht werden. Die bereits ausgearbeiteten Anforderungen stellen die Basis für die Bewertung der Produkte dar. Aufbauend darauf wird zwischen folgenden Erfüllungsgraden unterschieden:

- **„o“:** Die Anforderung wird nicht erfüllt.
- **„+“:** Die Anforderung wird nur teilweise oder nur über eine Produkterweiterung erfüllt.
- **„++“:** Die Anforderung wird erfüllt.
- **„+++“:** Die Anforderung wird mehr als erwartet erfüllt.

Eine Gewichtung der einzelnen Anforderungskriterien wird nicht erfolgen, da durch eine Wertung ein Ergebnis manipulierbar und zu sehr vom konkreten Einsatzszenario abhängig ist. Darüber hinaus stellt ein Gesamtergebnis zwar eine interessante Erkenntnis dar, wesentlich bedeutsamer sind in diesem Fall aber die Unterschiede innerhalb der einzelnen Kategorien.

Eine gute Form zur Visualisierung von gleichwertigen Kategorien ist ein Netzdiagramm. Bei diesem wird jede Anforderung einer Achse zugeordnet. Für alle Achsen gilt die gleiche Orientierung; die besseren Werte liegen einheitlich im Zentrum oder außerhalb der Strahlen. Die Achsen werden kreisförmig in 360 Grad gleichmäßig angeordnet. Die Werte jeder Serie werden mit Linien verbunden. Bei mehreren Serien werden verschiedene Farben verwendet. (vgl. Abbildung 4-6)

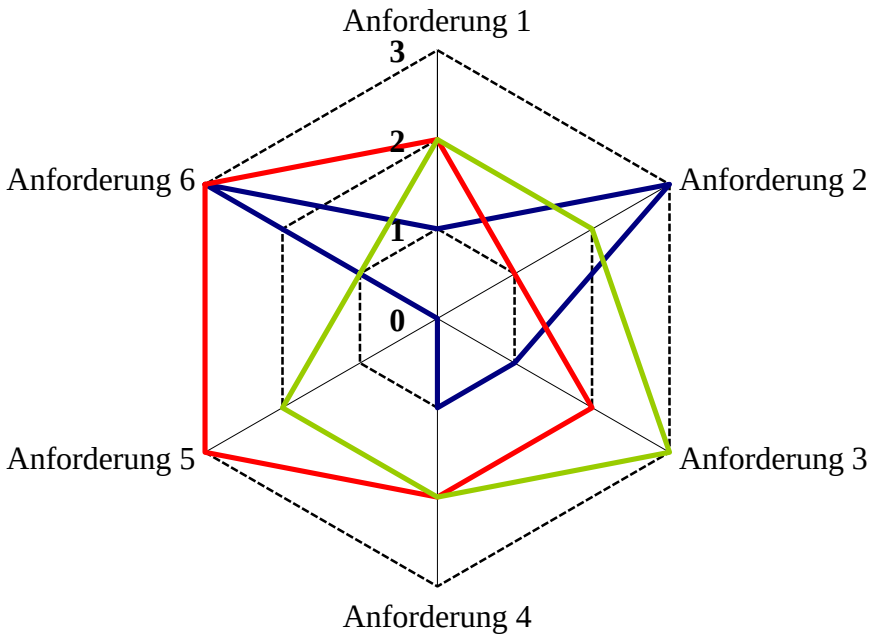


Abbildung 4-6: Netzdiagramm

Bei der großen Menge der Anforderungen geht die Übersichtlichkeit bei einem Netzdiagramm verloren. So ist die Entscheidung auf eine Bewertungsmatrix gefallen (siehe Tabelle 4-1), bei der die Visualisierung der Wertung in Form einer Tabelle erfolgt.Die Matrix besteht im Konkreten Fall aus vier Spalten; in der ersten Spalte werden die Anforderungen aufgelistet, wobei die einzelnen Kategorien exponiert angezeigt werden. Die drei weiteren Spalten bilden sich jeweils aus den Produkten und ihren Bewertungen. Am Ende der Tabelle wird ein Gesamtwert berechnet, bei dem die einzelnen „+“-Zeichen addiert werden.

Anforderung	Microsoft SharePoint Server 2010	IBM Lotus Connections	Atlassian Confluence
Allgemein	15	11	14
Ajax	++	++	+++
RSS/Atom	+++	++	+++
Kollaborationsbereiche	++	+	++
WYSIWYG-Editoren	+++	++	++
Mehrsprachigkeit	++	+	+
Globale Suche	+++	+++	+++
Blogs	16	18	14
Einträge	+	++	+
Kommentare	+	++	+
Verlinkung	++	++	++
Archiv	++	++	++
Kategorien und Tagging	++	+	+
Abonnement	+	+++	++
Suche	++	++	++
Rollenmanagement	+++	++	++
Verwaltung	++	++	+
Microblogs	2	4	3
Einträge	+	+	+
Vernetzung	+	+	+
Kommunikation	o	++	o
Tagging	o	o	o
Suche	o	o	+
Schnittstellen	o	o	o
Verwaltung	o	o	o
Wikis	19	18	26
Artikel	+++	++	+++
Qualitätssicherung	++	++	++
Übersichtsseite	++	+	+++
Verlinkung	++	++	+++
Kategorien und Tagging	++	+++	++
Abonnement	++	++	+++
Export	o	o	+++
Suche	++	++	++
Rollenmanagement	++	++	++
Verwaltung	++	++	+++
Social Networking Services	10	11	7
Profil	+++	++	+
Vernetzung	++	++	+
Activity Stream	++	+	++
Kommunikation	+	+	o
Identitätsverfolgung	++	+++	+++
Tagging	o	++	o
Social Tagging	6	6	6
Tagging	++	+	+
Anzeige von Tags	++	+++	+++
Social Bookmarking	++	++	++
Weitere Anwendungen	3	3	1
Foren	++	++	+
Instant Messaging	o	+	o
Gruppeneditoren	+	o	o
Collaborative Filtering	o	o	o
Summe	71	71	73

Tabelle 4-1: Bewertungsmatrix der Social Software-Kollaborationsplattformen

4.3 Auswertung

Der SharePoint Server 2010 liefert trotz seiner Herkunft ein gutes Ergebnis mit 71 Punkten. Es gibt zwar noch einige Schwächen, wie das Zuweisen von Tags bei Profilen oder das Exportieren von Wiki-Artikeln, trotzdem sind die wichtigsten Social Software-Funktionalitäten inzwischen so eingesetzt, dass diese die Zusammenarbeit um eine Vielfaches verbesser können.

Lotus Connections ist die Anwendung von den drei Produkten, die von Anfang an als reine Social Software-Kollaborationsplattform geplant war. Das schlägt sich auch auf das Ergebnis der Bewertung nieder und so ist es nicht verwunderlich, dass das IBM-Produkt eine Gesamtpunktzahl von 71 Punkten erreicht.

Atlassain verfolgt mit Confluence einen anderen Ansatz als Microsoft und IBM mit ihren Produkten. So bringt das Produkt ohne Erweiterungen bei vielen Funktionalitäten nur eine Grundausstattung mit. Dafür existiert aber eine große Anzahl von Plug-ins, darunter auch kostenlose, im Internet, die viele der untersuchten Komponenten noch weiter verbessern. Davon, dass Confluence die meisten Punkte bei den Wiki-Funktionalitäten holt, war auszugehen. Mit 73 Gesamtpunkten wurde bei diesem Vergleich der erste Platz erreicht.

Allgemein ist zu erkennen, dass die Integration der Social Software bereits weit vorangeschritten ist. Schwächen, wie Stärken haben die bewerteten Produkte alle. Auffällig ist jedoch, dass alle beim noch recht jungen Social Software-Anwendungstyp Microblog Probleme haben. Es werden nur benutzerbezogene Microblogs ermöglicht, die zumindest bei Confluence und SharePoint nichts mehr als Statusaktualisierungen sind. Hier besteht eindeutig das größte Verbesserungspotenzial.

5 Optimierung

Die Optimierung sollte, wie bereits ausgeführt am Microsoft Sharepoint Server 2010 stattfinden. Im letzten Kapitel wurden einige Funktionalitäten und Eigenschaften aufgezeigt, die verbessert werden können. Zeitlich gesehen ist eine komplette Abarbeitung dieser Punkte nicht realisierbar. Die größten Defizite besitzt der SharePoint Server bei der Einbindung einer Microblog-Anwendung. Kurzum gesagt, es existiert keine wirkliche Microblog-Anwendung im Sharepoint. Hier soll die folgende Programmierung ansetzen.

5.1 Technologische Voraussetzungen

Um eine programmatische Lösung des Problems angehen zu können, müssen wichtige technologische Grundlagen für die Entwicklung und die Anforderungen an die Implementierung erarbeitet werden.

5.1.1 Entwicklungsumgebung

Mit der neuesten Version der Entwicklungsumgebung Visual Studio (Visual Studio 2010) wurden wichtige Funktionalitäten für die Entwicklung, sowie umfangreiche Projekt- und Elementvorlagen für den SharePoint Server 2010 fest integriert. (vgl. Zahorsky 2010) Visual Studio wurde deshalb in der virtuellen Maschine, die bereits für die Evaluierung genutzt wurde, installiert. Außerdem hat die Konstellation Visual Studio und SharePoint Server auf der gleichen Maschine den Vorteil, dass ein Testen der entwickelten Software direkt aus der Entwicklungsumgebung heraus möglich ist.

Weiter ist in der T-Systems Multimedia Solutions ein Team Foundation Server von Microsoft im Einsatz, der auch bei der Bearbeitung der Implementierung verwendet werden soll. Mit Hilfe des Servers können Projektdaten für ein Team zentral gespeichert und verwaltet werden. (vgl. Schwichtenberg 2010) Vor allem die Funktio-

nen zur Versionskontrolle sind auch für einzelne Entwickler interessant. Mit diesen ist das Ein- und Auschecken von Projekt-Dateien möglich, wobei Kommentare zu den Versionen hinterlegt werden können. Zusätzlich können ältere Versionen wiederhergestellt werden.

5.1.2 SharePoint-Entwicklung

Der SharePoint Server 2010 wurde auf Grundlage des .NET-Frameworks und ASP.NET 3.5 ServicePack 1 entwickelt. Implementierungen für den SharePoint Server müssen auf den gleichen Technologien basieren. Das .NET-Framework ist eine Plattform für die Anwendungsentwicklung. Das Framework beinhaltet über 8000 Objekte (Klassen) und unzählige wiederverwendbare Funktionen.

.NET ermöglicht die Programmierung mit verschiedenen Programmiersprachen. Die im Visual Studio 2010 integrierten Projektvorlagen für den SharePoint Server sind entweder in C# oder Visual Basic (VB.NET) verfügbar. Für die Implementierung wurde die Programmiersprache C# gewählt, weil die meisten Tutorials und Projekte für den SharePoint Server, die im Internet zugänglich sind, in C# geschrieben sind.

5.1.2.1 ASP.NET

ASP.NET ist die auf .NET basierende Bibliothek für die Programmierung von Webseiten und Webdiensten. In ASP.NET-Webseiten findet bei der Benutzeroberflächenprogrammierung eine klare Trennung der visuellen und der logischen Komponente statt. Es wird zwischen dem sichtbaren Teil einer Seite und dem Code hinter der Seite, der mit den Seitenelementen interagiert, unterschieden.

Die visuelle Komponente besteht aus einer Datei (typische Endung „.aspx“), die statisches Markup wie HTML und/oder ASP.NET-Serversteuerelemente (Web

Controls) enthält. Die ASP.NET-Webseite fungiert als Container für den statischen Text und die Steuerelemente, die angezeigt werden sollen.

Die ASP.NET-Webserver-Steuerelemente sind Objekte auf ASP.NET-Webseiten, die beim Anfordern der Seite ausgeführt werden und Markup für einen Browser ausgeben. Viele Webserver-Steuerelemente ähneln bekannten HTML-Elementen, z.B. Schaltflächen und Textfeldern. Andere Steuerelemente umfassen komplexere Verhaltensweisen, z.B. Kalender-Steuerelemente und Steuerelemente, mit denen Datenverbindungen verwaltet werden. (vgl. Microsoft 2010c)

Zusätzlich zu Webserver-Steuerelementen auf den ASP.NET-Webseiten können eigene, benutzerdefinierte, wiederverwendbare Steuerelemente erstellt werden, indem die gleichen Verfahren wie beim Erstellen von ASP.NET-Webseiten verwendet werden. Diese Steuerelemente werden Benutzersteuerelemente (User Controls) genannt. Ein Benutzersteuerelement stellt eine Art zusammengesetztes Steuerelement dar, das fast wie eine ASP.NET-Webseite funktioniert – so können einer User Control vorhandene Webserver-Steuerelemente und Markup hinzugefügt, sowie Eigenschaften und Methoden für das Steuerelement definiert werden. Die Benutzersteuerelemente können anschließend in ASP.NET-Webseiten eingebettet werden, wo sie als Einheit fungieren. (vgl. Microsoft 2010d) Die typische Datei-Endung ist „.ascx“.

Der Code, der für die Interaktion mit den Seiten dient, kann entweder in einen Skript-Block in der Seite oder in einer separaten Klasse eingefügt werden. Wenn sich der Code in einer separaten Klassendatei befindet, wird diese Datei als Code-Behind-Datei bezeichnet.

ASP.NET-Webseiten werden in eine DLL-Datei (Dynamic Link Library) kompiliert. Wenn ein Benutzer das erste Mal eine ASPX-Seite aufruft, generiert ASP.NET automatisch eine .NET-Klassendatei, die die Seite darstellt. Diese Datei wird dann kompiliert. Die DLL-Datei wird auf dem Server ausgeführt und erzeugt dynamisch die HTML-Ausgabe für die Seite. (vgl. Microsoft 2010e)

5.1.2.2 Webparts

ASP.NET-Webparts stellen einen integrierten Steuerelementsatz zum Erstellen von Websites dar, mit denen Endbenutzer den Inhalt, die Darstellung und das Verhalten von Webseiten direkt in einem Browser ändern können. Die Änderungen gelten für alle Benutzer der Website oder für einzelne Benutzer. Wenn Benutzer Webseiten und Steuerelemente ändern, können diese Einstellungen gespeichert werden. Auf diese Weise erhalten Benutzer die Möglichkeit, ihre persönlichen Einstellungen für zukünftige Browsersitzungen beizubehalten. Diese Option wird auch als Personalisierung bezeichnet. Mithilfe dieser Webparts-Funktionen ermöglichen Entwickler Endbenutzern das dynamische Personalisieren von Webanwendungen, ohne dass Aktionen seitens des Entwicklers oder eines Administrators erforderlich sind. (vgl. Microsoft 2010f)

Webparts können im SharePoint in Webpart-Seiten über Webpart-Zonen, festgelegte Bereiche innerhalb einer Seite, eingebunden werden. Eine Ausnahme bilden Wiki-Seiten im SharePoint Server 2010, dort sind die Webparts an jeder Stelle innerhalb eines Artikels einfügbar. (siehe Abbildung 5-1)

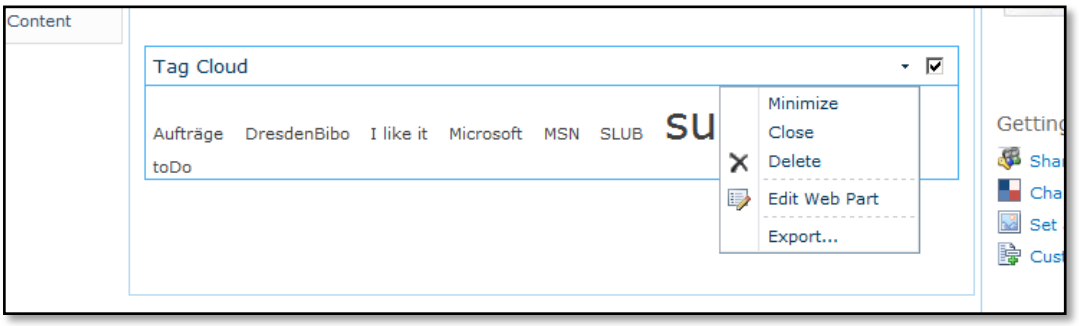


Abbildung 5-1: Tag Cloud-Webpart innerhalb eines Wiki-Artikels

5.1.2.3 Features

Im SharePoint-Jargon handelt es sich bei einem Feature um XML-Definitionen und Dateien („.aspx“, Grafiken etc.), die bestimmte SharePoint-Erweiterungen wie Webparts, Workflows, Event Handler, Listen, Datenfelder (Site oder List Columns)

beschreiben. Features lassen sich nach Installation für bestehende Websites – alternativ können Features auch auf Farm-, Webanwendungen- und Websitesammlungen-Ebene eingesetzt werden – von den Administratoren aktivieren und bei Bedarf auch wieder deaktivieren. (vgl. Microsoft 2010g) Visual Studio 2010 bietet für Features einen eigenen Designer, so ist es nicht nötig das genaue XML Schema zu kennen.

5.1.2.4 SharePoint-Solution

Um unter anderen Features für einen SharePoint Server bereitzustellen, muss eine sogenannte SharePoint Solution erstellt werden. Es handelt sich dabei um ein Installationspaket im CAB-Format. Die Datei-Endung ist „.wsp“ – abgeleitet von Web Solution Package – eine Bezeichnung die alternativ verwendet wird. (vgl. Microsoft 2010h)

Eine Solution kann unter anderen mit Hilfe der Windows PowerShell einem SharePoint Server-Farm hinzugefügt werden. Dort muss sie dann für jede Webanwendung einzeln bereitgestellt werden. Dazu kommen folgende Befehle zum Einsatz:

```
Add-SPSolution C:\Microblog.wsp #fügt die Solution dem Server hinzu
Install-SPSolution –Identity Microblog.wsp –WebApplication http://sp2010.de –
   GACDeployment #aktiviert die Solution für die http://sp2010.de-Webanwendung
```

5.2 Anforderungen

Die allgemeinen Anforderungen an eine Microblog-Anwendung wurden bereits im Kapitel 3.2.2 aufgeführt. Da die Zeit für die Entwicklung dieser Anwendung aber stark begrenzt ist, wurden folgende abgewandelte Anforderungen für die Implementierung definiert:

Als erste Anforderung sollen mehrere Microblogs pro Websitesammlung möglich sein, weil mit den Activity-Feeds und den Statusaktualisierungen genügend Mög-

lichkeiten zur Verfügung stehen, um benutzerzentrierte Nachrichten zu streuen. Es soll eine Lösung entwickelt werden, mit der Microblogs für Teams, Projektgruppen, etc. eingerichtet werden können.

Die Einträge müssen auf eine bestimmte Zeichenanzahl begrenzt werden. Die Nutzer sollen sich neben den Beiträgen eines Microblogs, alle Einträge aller Microblogs, auf denen sie Zugriff haben, anzeigen lassen können. Gleichzeitig sollte eine Ansicht angeboten werden, mit deren Hilfe sich die Nutzer ihre eigenen Beiträge aller Microblogs darstellen und entfernen lassen können. Die Nutzer dürfen nicht die Einträge der anderen Anwender löschen.

Es sollten Einträgen Tags zugewiesen werden können und die Möglichkeit einer Darstellung als Tag Cloud bestehen. Einträge als gezielte Nachrichten an bestimmte Nutzer hervorzuheben und eine Suchfunktion sind wichtige Anforderungen.

Zu den optionalen Anforderungen gehört die Unterstützung eines mehrsprachigen User Interface, wobei auf die Mehrsprachigkeits-Funktionalität des SharePoint Servers zurückgegriffen werden soll. Auch nützlich wäre die Auswahl von Benachrichtigungen per E-Mail von bestimmten Microblogs aus, sowie die Integration der SharePoint-Suche.

5.3 Zeitliche Planung

Nach der Bewertung der einzelnen Produkte standen noch zwei Monate zur Verfügung. Dabei wurden eineinhalb Monate zur Programmierung geplant und zwei Wochen für die Dokumentation der Umsetzung.

Darüber hinaus wurde die Aufgabe so gelöst, dass die ersten zwei Wochen der Programmierung für das Gerüst der Anwendung und zum Testen allgemeiner funktioneller Fragen genutzt wurden. Die darauf folgenden zwei Wochen wurden für die Umsetzung der wichtigen bzw. notwendigen Funktionen angesetzt. Die schließlich

verbleibende Zeit wurde zum Einbinden optionaler Bestandteile und zum Testen veranschlagt.

5.4 Implementierung

Die fertige Microblog-Anwendung sieht wie in Abbildung 5-2 aus. Die einzelnen Bestandteile und die allgemeine Struktur der Anwendung werden im folgenden Kapitel erklärt.

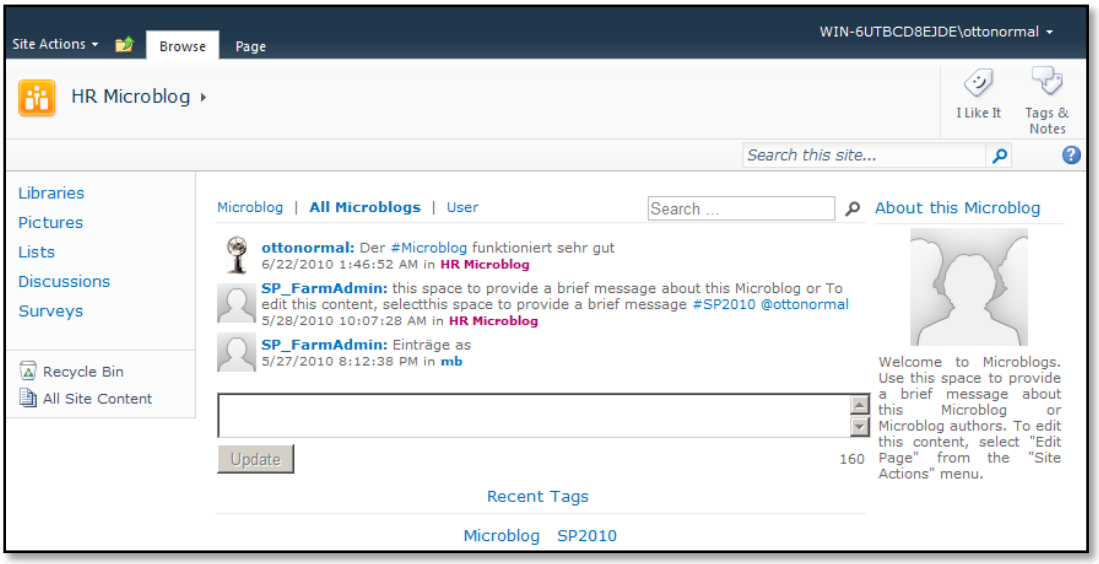


Abbildung 5-2: Microblog-Anwendung im SharePoint Server

Die meisten Dateien der Anwendung wurden mit Hilfe der Visual Studio-Vorlagen für SharePoint erstellt. Bei diesen musste zum Großteil nur geringe Veränderungen vorgenommen werden. Der Hauptteil der Programmierung erfolgte bei der Erstellung der User Control des Visual Webparts, deren Quellcode im folgenden Kapitel öfters zitiert wird, weshalb dieser im Anhang eingefügt wurde.

Die Basis der fertigen Solution (*MicroblogSolution*) bildet ein Site Definition-Projekt (*Microblog*). Das Solution-Paket setzt sich dementsprechend aus einer Site Definition, einem Feature – zu dem ein Visual Webpart und eine List Definition gehört – Ressource-Dateien und einem Bild zusammen. (siehe Abbildung 5-3)

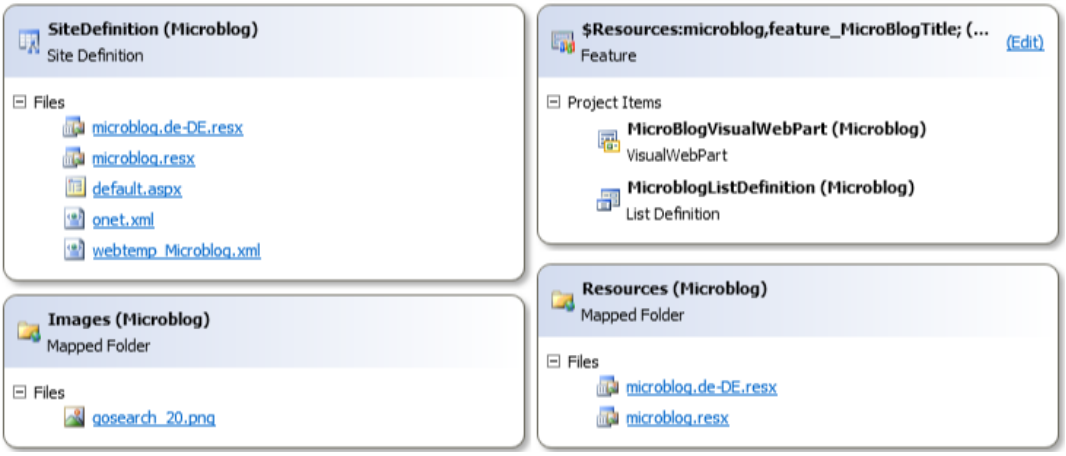


Abbildung 5-3: MicroblogSolution-Paket

5.4.1 Die Websitevorlage

Nachdem die *MicroblogSolution* für eine Webanwendung bereitgestellt wurde, kann bei der Erstellung einer neuen Website, wie in Abbildung 5-4 zu sehen ist, eine Microblog-Website ausgewählt werden.

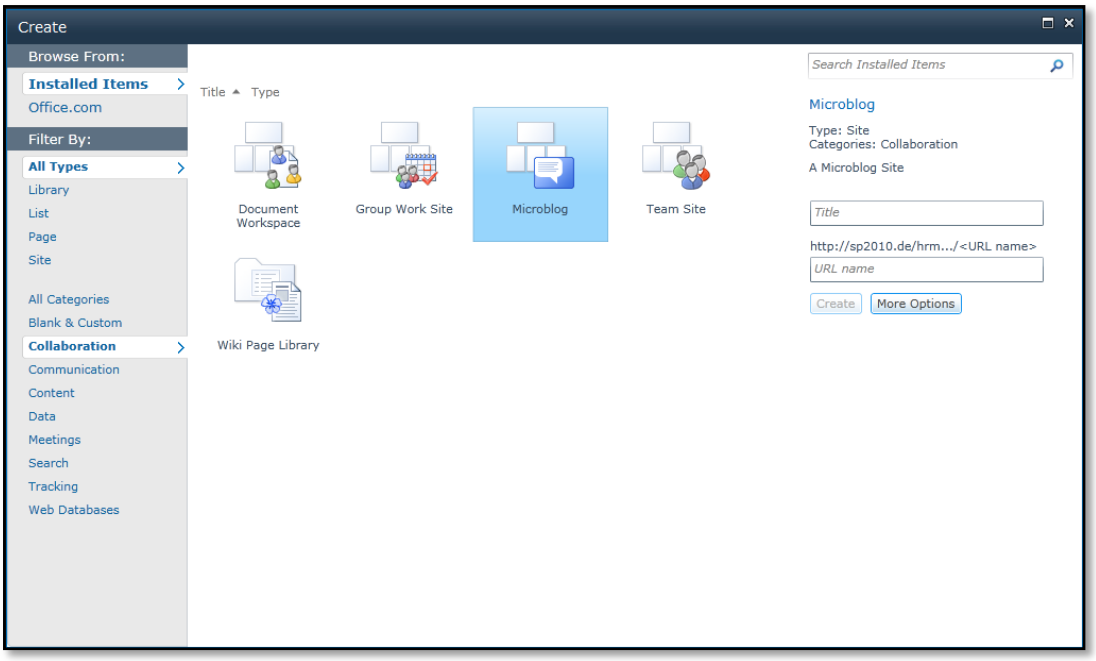


Abbildung 5-4: Erstellung einer neuen Microblog-Website

Die bei der Erstellung angezeigten Eigenschaften (Bild, Kategorie, etc.) wurden über die *webtemp_Microblog.xml*-Datei realisiert. Diese Datei ist für die Registrierung und Bereitstellung im SharePoint Server zuständig.

Die verschiedenen Konfigurationen und Module der Website, die mit Hilfe der Vorlage erstellt wird, werden über die *onet.xml*-Datei festgelegt. Für die Microblog-Websitevorlage wurde eine Standardkonfiguration (*Microblog*) festgelegt, der ein Modul (*DefaultBlank*) zugewiesen ist.

In dem Konfigurationsteil der XML-Datei wird angewiesen, dass eine Microblog-Liste bei der Website-Erzeugung erstellt wird. In dieser sollen später die Microblog-Einträge gespeichert werden. Weiter wird ein sogenanntes *TeamCollab*-Feature für die Website aktiviert. Das ermöglicht unter anderen, dass später verschiedene Bibliotheken und Listen innerhalb der SharePoint-Website erstellt werden können.

```
<Configurations>
  <Configuration ID="0" Name="Microblog">
    <Lists>
      <List FeatureId="0029230e-67a8-4a79-bfe1-ca22049b98e8" Type="10133"
         Title="Microblog List"
         Url="$Resources:core,lists_Folder;/MicroblogList"/>
    </Lists>
    …
    <WebFeatures>
      <!-- TeamCollab Feature -->
      <Feature ID="00BFEA71-4EA5-48D4-A4AD-7EA5C011ABE5" />
    </WebFeatures>
    <Modules>
      <Module Name="DefaultBlank" />
    </Modules>
  </Configuration>
</Configurations>
```

Das Modul-Element legt Dateien oder Dateisammlungen und den Ort, wo diese bei der Website-Erstellung installiert werden, fest. Im *DefaultBlank*-Modul ist das nur die *default.aspx*-Datei. Dabei handelt es sich um eine Webpart-Seite. Zusätzlich wird im Modul definiert, ob Webparts und in welche Webpart-Zone diese geladen werden.

Es können also, wie in Abbildung Abbildung 5-5 zu sehen, in einer Websitesammlung mehrere Microblog-Websites, die jeweils eine Microblog-Liste und eine Webpart-Seite beinhalten, angelegt werden.

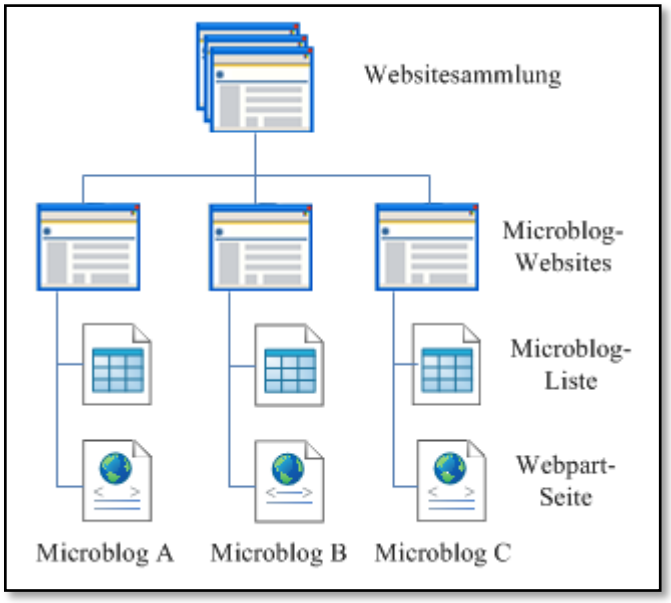


Abbildung 5-5: Struktur Microblog-Websites

5.4.2 Die Webpart-Seite

In der *default.aspx* befinden sich diverse Layout-Angaben für die angezeigt Seite. Der wichtigste Abschnitt der Datei ist ein *table*-Element, in dem zwei Webpart-Zonen (*Left* und *Right*) festgelegt werden:

```
<table cellpadding="0" cellspacing="0" style="width: 100%; padding: 5px 10px 10px
   10px;">
  <tr>
    <td valign="top">
      <WebPartPages:WebPartZone runat="server" FrameType="TitleBarOnly" ID="Left"
         Title="loc:Left" AllowPersonalization="false" />
    </td>
    <td valign="top" class="ms-blogRightZone" style="width: 165px; word-wrap:
         break-word; overflow-x: hidden;">
      <WebPartPages:WebPartZone runat="server" FrameType="TitleBarOnly"
         ID="Right" Title="loc:Right" AllowPersonalization="false" />
      …
    </td>
  </tr>
</table>
```

Im *DefaultBlank*-Modul der Websitevorlage wurde festgelegt, dass in die Webpart-Zone *Right* ein Content-Webpart, der zum Repertoire jedes 2010er SharePoint Servers gehört, geladen wird. Er ermöglicht, dass ein Administrator ein Bild und eine kurze Beschreibung für den Microblog angeben kann.

In die Webpart-Zone *Left* wird ein eigens entwickelter Webpart (*MicroBlogVisualWebPart*) geladen. Er stellt die eigentliche Microblog-Anwendung dar. Nach der Erstellung können weitere Webparts in die Zonen eingefügt werden. (siehe Abbildung 5-6)

Abbildung 5-6: Webpart-Zonen

5.4.3 Die Microblog-Liste

Für die Microblog-Liste (*Microblog List*), in der die Einträge gespeichert werden, wurde eine eigene Listenvorlage (*MicroblogListDefinition*) angelegt. Diese besteht aus den Dateien *Schema.xml* und *Elements.xml*. Letztere wird unteranderen benötigt, falls eine neue Liste über die SharePoint-Benutzeroberfläche erstellt wird. Diese Funktion ist für die Lösung nicht angedacht. Es wurde festgelegt, dass die Listenvorlage und die erstellten Listen für die Nutzer nicht sichtbar sind.

In der *Schema.xml* wird definiert, welche Ansichten, Formulare, Symbolleisten und Listenfelder zu einer *SharePoint*-Liste gehören. Jeder Listenvorlage muss ein Listentyp zugewiesen werden, der diverse Eigenschaften vererbt. Für die Microblog-

Liste wurde der *Custom List*-Typ ausgewählt. Zusätzlich wurden zwei weitere Felder eingerichtet:

```
<Fields>
  <Field ID="{EFFC5AAC-F252-4C02-ABEC-ABFF77819057}" Type="Text" Name="Tweet"
    DisplayName="Tweet" StaticName="Tweet" MaxLength="160"></Field>
  <Field ID="{EE046662-E4E7-4B94-916E-7761CE13C2CC}" Type="Text" Name="HashTags"
    DisplayName="HashTags" StaticName="HashTags"></Field>
</Fields>
```

Im Listenfeld *Tweet* wird der Eintragstext gespeichert, der auf 160 Zeichen begrenzt ist. In *HashTags* werden die zu einem Eintrag gehörenden Tags eingetragen. Inklusive der durch den Custom List-Typ vererbten, relevanten Felder ergibt sich folgende Liste:

Abbildung 5-7: Microblog-Liste

5.4.4 Das Microblog-Webpart

Das Herzstück der Microblog-Anwendung ist der *MicroBlogVisualWebPart*, welcher unter Benutzung der im SharePoint Server 2010 neu eingeführten Visual Webpart-Vorlage realisiert wurde. Bei dieser handelt es sich um einen Webpart, dem eine User Control zugewiesen ist. Vorteil bei der Variante ist, dass die Trennung zwischen Layout-Datei (*MicroBlogVisualWebPartUserControl.ascx)* und Code-Behind-Datei (*MicroBlogVisualWebPartUserControl.ascx.cs*) des Benutzersteuerelements den zu schreibenden Code reduziert. Zusätzlich steht im Visual Studio ein Design-Tool zur Verfügung, dass die Bearbeitung der visuellen Komponente vereinfacht.

In der Webpart-Datei (*MicroBlogVisualWebPart.cs*), in der die User Control festgelegt wird, wurden zusätzlich für den Webpart drei benutzerdefinierte Eigenschaften definiert. Ein Attribut davon regelt die Anzahl der angezeigten Einträge bis zum „more“-Link. Das zweite ist zum Regulieren der vergangenen Wochen, die für die Anzeige der Tags in der Tag Cloud einbezogen werden sollen. Und das dritte gibt ein Limit für die dargestellten Tags an. Diese Justierungen können über die Webpart-Einstellungen im Browser eingegeben werden, wie in Abbildung 5-6 zu sehen ist. Sie gelten nur für den jeweiligen Microblog.

5.4.4.1 Microblog-Einträge

Die Textbox und der Update-Button (siehe Abbildung 5-8) im Webpart dienen den Nutzern dazu neue Einträge abzuschicken. Rechts unterhalb des Eingabefelds wird dem Nutzer angezeigt, wie viele Zeichen bei der Eingabe noch übrig sind, damit das Limit von 160 Zeichen nicht überschritten wird. Wird dieser Wert überstiegen, wird in roter Farbe die Anzahl der überschrittenen Zeichen dargestellt und der Update-Button deaktiviert. Dabei kommen verschieden JavaScript-Funktionen zum Einsatz (Zeile 21). Die Verwendung von JavaScript wurde jedoch versucht gering zu halten.

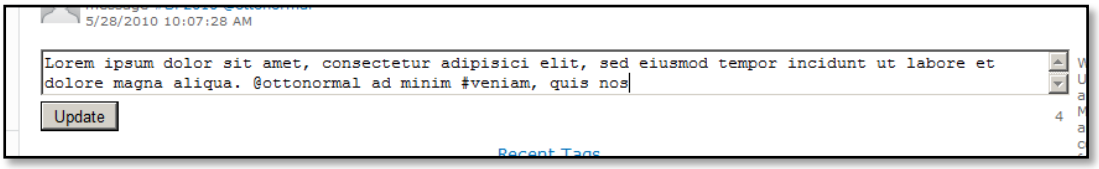


Abbildung 5-8: Eingabe eines Microblog-Eintrags

Mitten im Text können mehrere Tags eingefügt werden. Dabei wurde die, aus Twitter bekannte, „#“-Zeichen-plus-Tag-Form übernommen. Außerdem können auch Nutzer angegeben werden, für welche der Post interessant sein könnte. Dazu wird ein „@“-Zeichen plus der im SharePoint bekannte Benutzername eingetragen.

Die Einträge werden dann über die Funktion *setTweet()* (Zeile 862) in die zum Microblog gehörende Liste eingetragen. Die Hashtags werden unter Benutzung der

Funktion *getHashTags()* (Zeile 897) aus dem Eintrag ausgelesen und in das *HashTags*-Feld gespeichert. Dieser Ansatz wird verfolgt, damit eine spätere Integrierung des Social Tagging-System des SharePoints einfacher möglich ist.

5.4.4.2 Navigation

Nicht nur die Darstellung des Microblogs wird über den Webpart verwirklicht, sondern es stehen über die Registerkarten (engl. Tab) weitere Ansichten zur Verfügung:

Microblog-Tab

Der Microblog-Tab wird nach jedem neuem Laden des Webparts automatisch ausgewählt. Dabei werden die Einträge des derzeit besuchten Microblogs chronologisch aufgelistet. Am Anfang jedes Posts wird das Profilbild des jeweiligen Nutzers abgebildet, daneben werden Benutzername, Eintragstext und Erstellungsdatum angezeigt. (siehe Abbildung 5-9) Umgesetzt wird die Darstellung im Programmcode durch die Funktion *getMicroblogTweets()* (Zeile 299).

Abbildung 5-9: Microblog-Tab

Beim Benutzername, der auf die entsprechende Profilseite verweist, bei den Hashtags und bei den „@“-Zeichen-plus-Benutzername-Zeichenketten handelt es

sich um Links, die dynamisch nach dem Auslesen der Microblog-Liste erzeugt werden.

Bei der Ausführung einer ASP.NET-Seite durchläuft diese Seite einen Lebenszyklus, der sich aus einer Reihe von Verarbeitungsschritten zusammensetzt. Dazu zählen unter anderem die Initialisierung, das Instanziieren von Steuerelementen, das Wiederherstellen und Aufrechterhalten des Zustands, das Ausführen von Event Handler-Code und das Rendern. Die Zuweisung von dynamisch erzeugten Event Handler in User Controls ist kompliziert und dank des Lebenszyklus nicht für selbst erzeugte Elemente an bestimmten Stellen möglich.

Dieses Problem wurde unter Benutzung des Repeater-Steuerelements umgangen. Innerhalb des Steuerelements werden Templates für die einzelnen Elemente eines Microblog-Eintrags angegeben. In der *getMicroblogTweets()*-Funktion (Zeile 299) wird dann eine DataTable mit den Einträgen erzeugt (Zeile 330), die dem Repeater als Datenquelle zugewiesen wird (Zeile 392). Dadurch werden die zu rendernden Elemente automatisch erzeugt und automatisch die Event Handler festgelegt.

Ab der, in der Webpart-Einstellung zugewiesenen Anzahl von Einträgen, wird zwischen den Posts und der Eingabe-Textbox ein Link mit dem Text „more“ (in dt. Lokalisierung „mehr“) angezeigt. Wird dieser betätigt, dann werden dementsprechend mehr Einträge aufgelistet.

Alle Microblogs-Tab

In dieser Ansicht werden alle Einträge aller Microblogs chronologisch aufgelistet, auf die der Nutzer Zugriff hat. Die Darstellung der Einträge unterscheidet sich dahingehend, dass zusätzlich ein Link zur Herkunfts-Microblog-Website angezeigt wird. (siehe Abbildung 5-10)

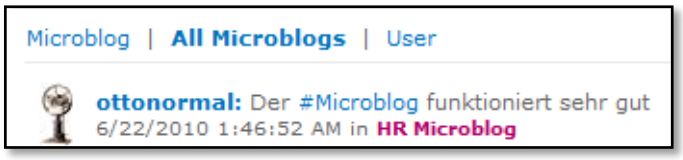


Abbildung 5-10: Alle Microblogs-Tab-Eintrag

Durch die *getAllTweets()*-Funtkion (Zeile 406) wird die Darstellung realisiert. Dabei werden alle Microblog-Websites der Websitesammlung abgefragt und überprüft, ob der Benutzer auf diese Zugriffsrechte hat.

Benutzer-Tab

Unter Verwendung der *getUserTweets()*-Funktion (Zeile 535) werden alle vom angemeldeten Nutzer veröffentlichten Einträge aufgelistet. Zu diesen kommen noch die Einträge dazu, die andere Anwender an den Nutzer gerichtet haben. Bei dieser Ansicht werden zusätzlich zum Link der Herkunfts-Microblog-Website ein „x“ neben dem Eintrag angezeigt. (siehe Abbildung 5-11) Über dieses können die Nutzer ihre bereits veröffentlichen Posts wieder löschen (*deletePost()* Zeile 1238).

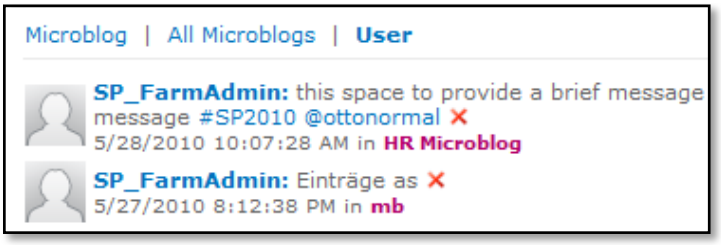


Abbildung 5-11: Benutzer-Tab-Einträge

5.4.4.3 Suche

Die Suche wird abhängig vom aktivierten Navigations-Tab über die Funktion *getSearchTweets()* (Zeile 681) aufgerufen. D.h. die gleichen Microblogs bzw. Einträge, die in dem ausgewählten Tab angezeigt werden, werden auch durchsucht. Als

Suchergebnis werden alle Einträge, die einen Treffer beinhalten, aufgelistet, wobei das Suchergebnis fett geschrieben ist. (siehe Abbildung 5-12)

Abbildung 5-12: Microblog-Suche

5.4.4.4 Tag Cloud

Am unteren Ende des Microblog-Webparts wird eine Tag Cloud angezeigt. (siehe Abbildung 5-13) Diese setzt sich genauso wie bei der Suche, je nachdem, welcher Navigations-Tab ausgewählt wurde, aus den eingetragenen Hashtags zusammen. Voreingestellt werden die Tags der letzten acht Wochen durchsucht.

Abbildung 5-13: Microblog-Tag Cloud

Dargestellt wird die Tag Cloud über die Funktion *generateTagCloud()* (Zeile 46). Die Tags werden alphabethisch sortiert und je nach Häufigkeit in vier unterschiedlichen Schriftgrößen dargestellt. Für die Größen wurden die CSS-font-size-Schlüsselwörter „small“, „medium“, „large“ und „x-large“ verwendet. Der Vorteil gegenüber festgelegten Längenangaben ist, dass sich allgemeine Anpassungen für eine Websitesammlung bzw. Website auch auf das Webpart ausschlagen. So passt die Anwendung grafisch besser zum Gesamtkonzept, ohne dass spezielle Änderungen notwendig sind. Dieses Konzept wird allgemein bei allen Layout-Elementen verfolgt.

Bei der Vergabe der Schriftgrößen wird eine logarithmische Funktion verwendet, um bei einer ungleichmäßigen Verteilung der Tags eine möglich ausgewogene Darstellung zu erreichen. Wird ein Tag in der Wortwolke oder in einem Eintrag ausgewählt, wird eine Suche mit dem Tag als Suchbegriff ausgeführt.

5.4.4.5 Ajax

Ein zentraler Bestandteil für die Verwendung von Ajax in ASP.NET ist das UpdatePanel-Steuerelement. Es werden mehrere dieses Steuerelements verwendet, um die verschiedene Bereiche des Webparts unabhängig voneinander zu aktualisieren. Wenn die User Control zuerst gerendert wird, wird der Inhalt aller UpdatePanel-Steuerelemente gerendert und an den Browser gesendet. Bei nachfolgenden asynchronen Postbacks wird der Inhalt der einzelnen UpdatePanel-Steuerelemente in Abhängigkeit von deren Einstellungen aktualisiert.

So wird zum Beispiel für das Label, das die Restzeichen bei der Eingabe eines neuen Eintrags anzeigt, ein UpdatePanel verwendet:

```
<asp:UpdatePanel ID="UpdatePanelTweetInputLeft" runat="server">
  <ContentTemplate>
    <asp:Label runat="server" id="tweetInputLeft"></asp:Label>
  </ContentTemplate>
  <Triggers>
    <asp:AsyncPostBackTrigger ControlID="tweetSubmit" EventName="Click" />
  </Triggers>
</asp:UpdatePanel>
```

Im ContenTemplate-Abschnitt wird festgelegt, welches Element aktualisiert werden soll. Im Triggers-Element werden die Elemente zugewiesen, die eine Aktualisierung initiieren dürfen.

5.4.4.6 Sicherheitskonzept

Das Sicherheitskonzept von SharePoint sieht vor, dass Code immer im Kontext des aktuell angemeldeten Benutzers ausgeführt wird. Diese Vorgehensweise verhindert

Missbrauch durch Nutzer bzw. durch schlechte Programmierung. Einige Befehle benötigen aber mehr Rechte, als der angemeldete Anwender besitzt, sind aber für das Programm unabdingbar. Ein Beispiel dafür ist die *AllWebs*-Eigenschaft des *SPSite*-Objekts. Diese gibt alle Websites einer Websitesammlung zurück und wird für das Auffinden der Microblogs benötigt. Würde ein Standardnutzer ein Webpart mit folgendem Code ausführen, würde er eine Fehlermeldung bekommen:

```
using(SPSite cSite = new SPSite("http://sp2010.de"))
{
    foreach (SPWeb web in cSite.AllWebs)
    {
        // beliebiger Code
    }
}
```

Dieses Problem kann mit dem Konzept der Impersonisation umgangen werden. Dahinter steckt die Idee, während der Ausführung des Codes, in die Rolle eines anderen Benutzers zu schlüpfen und damit bestimmte Codeteile auszuführen. Die Klasse *SPSecurity* bietet eine Methode *RunWithElevatedPrivileges*, mit der Code als SharePoint-Systemadministrator ausgeführt werden kann. Folgendes Codebeispiel soll das Prinzip veranschaulichen:

```
SPSecurity.RunWithElevatedPrivileges(delegate()
{
    // mit Administratoren-Rechten ausgeführter Code
});
```

Nur der Code innerhalb der anonymen Methode wird mit Administratoren-Rechten ausgeführt. Über den Nutzerkontext erstellte Objekte, müssen innerhalb der Methode neu erstellt werden, da diese sonst trotzdem keine volle Berechtigung haben.

5.4.5 Mehrsprachigkeit

Für die Unterstützung eines mehrsprachigen User Interface bietet der SharePoint Server ein eigenes Lokalisierungskonzept an. Die Grundlage dafür bilden sogenannte Ressource-Dateien. Das sind XML-Dateien, die die Datei-Endung „.resx“ besitzen. Jede Ressource ist eine Zeichenkette, die über einen statischen Namen

abgerufen werden kann. Zur Verdeutlichung dient folgender Ausschnitt aus der *microblog.de-DE.resx*-Datei:

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  …
  <data name="feature_MicroBlogTitle">
    <value>Microblog</value>
  </data>
  …
</root>
```

Für jede Lokalisierung muss eine eigene .resx-Datei erstellt werden, dabei müssen die gleichen Namen als Schlüssel vorhanden sein. Die Benennung einer Ressource-Datei unterliegt dem Schema: *[Ressource-Dateiname].[Culture].resx*. Die Culture-Zeichenketten sind von Microsoft festgelegt. Für die deutsche Sprache ist das zum Beispiel „*de-DE*“. Die Dateien, bei denen dieses Kürzel weggelassen wird, werden geladen, wenn keine bestimmte oder eine nicht vorhandene Lokalisierung ausgewählt wurde.

Für die Solution wurden insgesamt zwei Ressource-Dateien erstellt: *microblog.de-DE.resx* und *microblog.resx*. Beide sind jedoch jeweils zweimal im Package vorhanden. Dies hat damit zu tun, wo die Dateien zur Einbindung im Code auf dem Server lokal gespeichert sein müssen. Die Dateien im gemappten Ordner *Ressources* werden global für den SharePoint Server hinterlegt. Diese werden bei der Bereitstellung von Websites und Features benötigt. So kann zum Beispiel der Titel des Microblog-Features, der in der Website-Administration angezeigt wird, wie in Abbildung 5-14 lokalisiert werden:

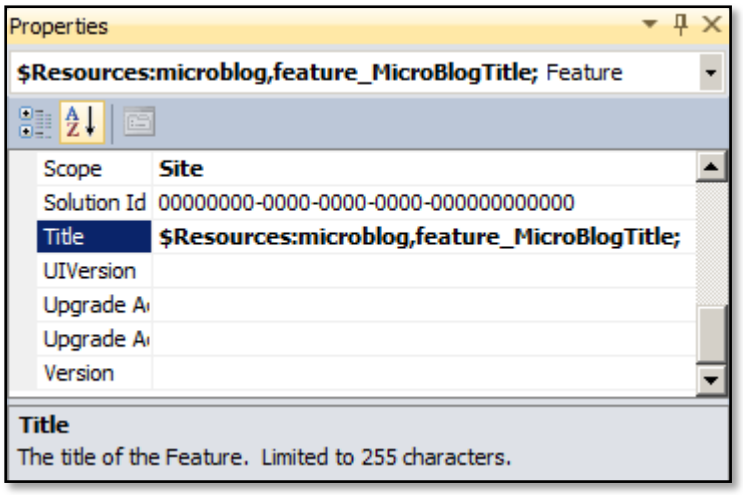


Abbildung 5-14: Feature-Titel

Die Resource-Dateien gehören zur Site Definition und sind damit dem Feature zugewiesen. Sie werden – über eine Einstellung zum Deploy-Ort – bei der Aktivierung des Features im Ressource-Ordner der jeweiligen Webanwendung gespeichert. Dieser Speicherort wird bei ASPX-Seiten und Assemblies zur Laufzeit benötigt. Ein Beispiel dafür ist die Einbindung, wie sie in der *MicroBlogVisualWebPartUserControl.ascx*-Datei realisiert wird – siehe Navigationslink „User“ Zeile 79-80:

```
<asp:literal runat="server" Text="<%$Resources:microblog,
   webpartnavigation_user%>" />
```

6 Zusammenfassung und Ausblick

Die Aufgabe bestand in der Konzeption, Analyse und Optimierung einer Social Software-Kollaborationsplattform und wurde erfolgreich umgesetzt. Während der Konzeption ist es gelungen Anforderungen an eine Social Software-Kollaborationsplattform auszuarbeiten, welche auf den Erkenntnissen der Forschungsgebiete CSCW und Enterprise 2.0 basieren. Die Umsetzung dieser Anforderungen wurde anschließend an ausgewählten Vertretern analysiert. Es reifte die Feststellung, dass die Einbindung von Social Software in den gewählten Produkten schon weit vorangeschritten ist. Es konnten aber auch Schwächen bei der Integration von Microblogs aufgezeigt werden.

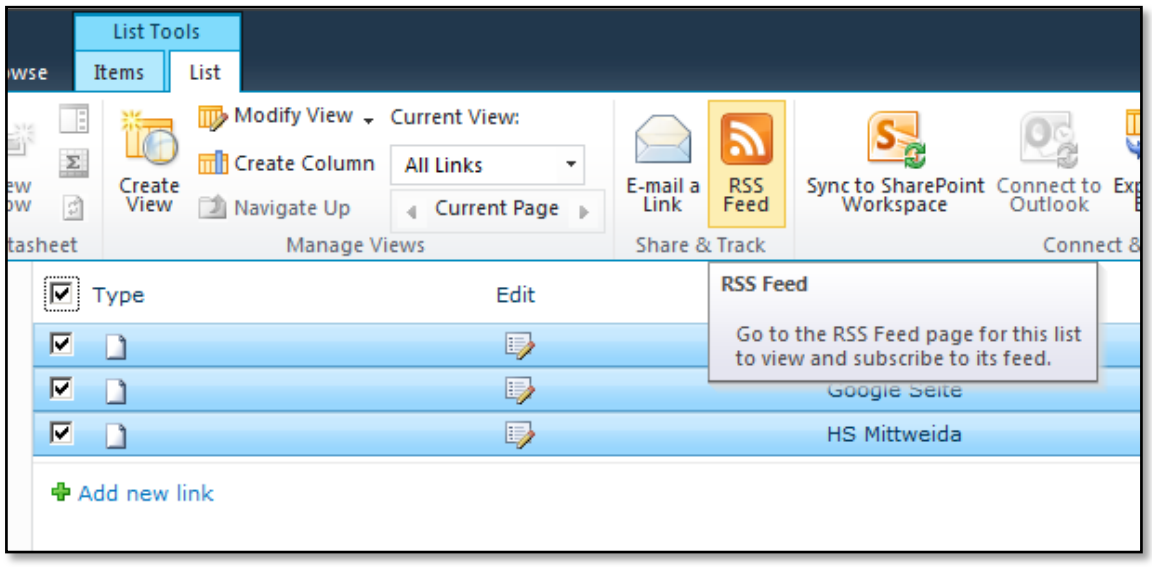
Auf Grundlage dieser Erkenntnis fand in der folgenden Optimierung eine Erstellung einer Microblog-Anwendung für den SharePoint Server 2010 statt. Alle an die Implementierung gestellten, wichtigen Anforderungen wurden realisiert. Auf Grund der zeitlichen Begrenzung konnten jedoch nicht alle Anforderungen an einen Microblog umgesetzt werden. Es wäre zu empfehlen diese offenen Punkte zu einem späteren Zeitpunkt zu beseitigen.

Darüber hinaus sind noch Änderungen für eine bessere Performance der Anwendung denkbar. So könnten die Abfragen der Listen über diverse Methoden zwischengespeichert werden. Es kann versucht werden, den Aufruf des User Profile-Dienstes zu meiden. Jeder Aufruf eines weiteren Dienstes verlangsamt den Server und die Anwendung. Auch die Integration in den SharePoint Server könnte weiter vorangetrieben werden, indem zum Beispiel die Social Tagging-Anwendung des SharePoints mit der Microblog-Anwendung verbunden wird.

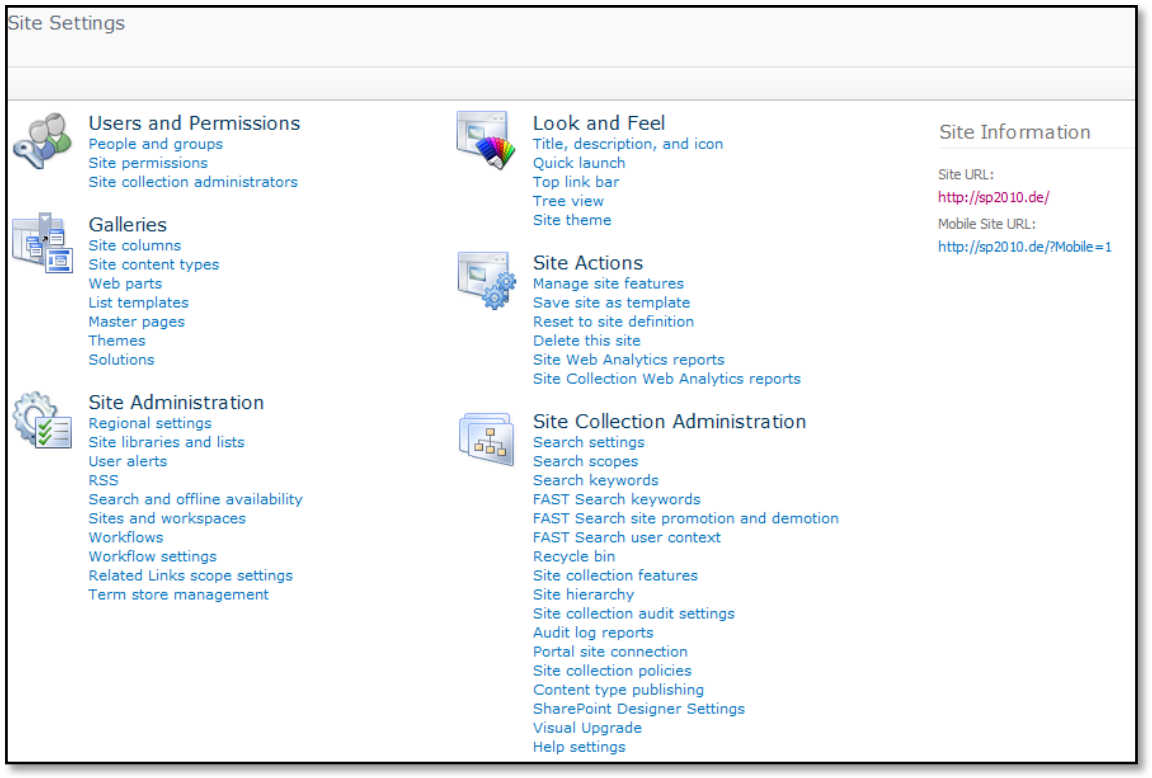
Allgemein ist anzumerken, dass die Entwicklungen rund um das Internet und Mobilität immer weiter voranschreiten. Konsequent wird bereits versucht ein Teil dieser Entwicklungen unter dem Begriff Web Squared zusammenzufassen. (vgl. O'Reilly und Battelle 2009) So sollte auch durch die vermehrte Verbreitung von Smartphones die fortschreitende Verzahnung mit den Kollaborationsplattformen ein wichtiges Thema sein.

Anhang

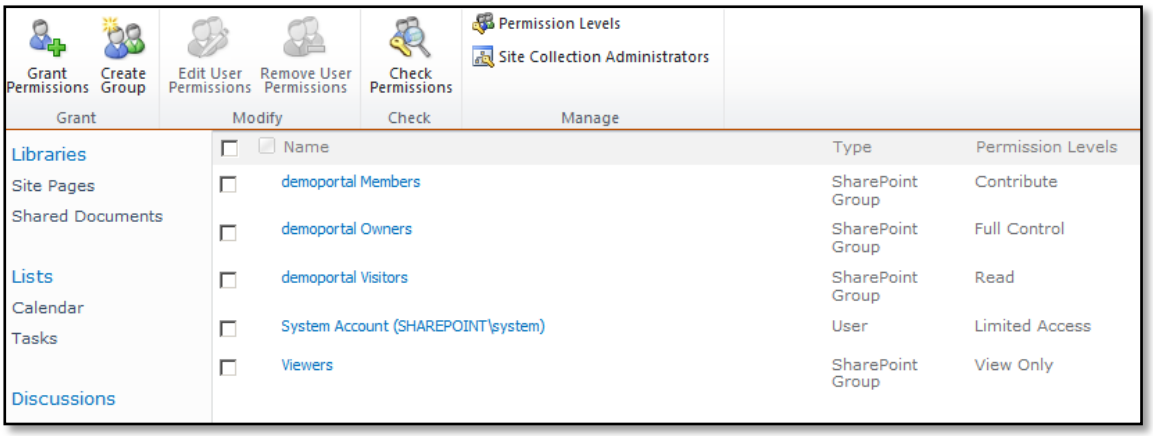
Anhang A – Bilder Microsoft SharePoint Server 2010



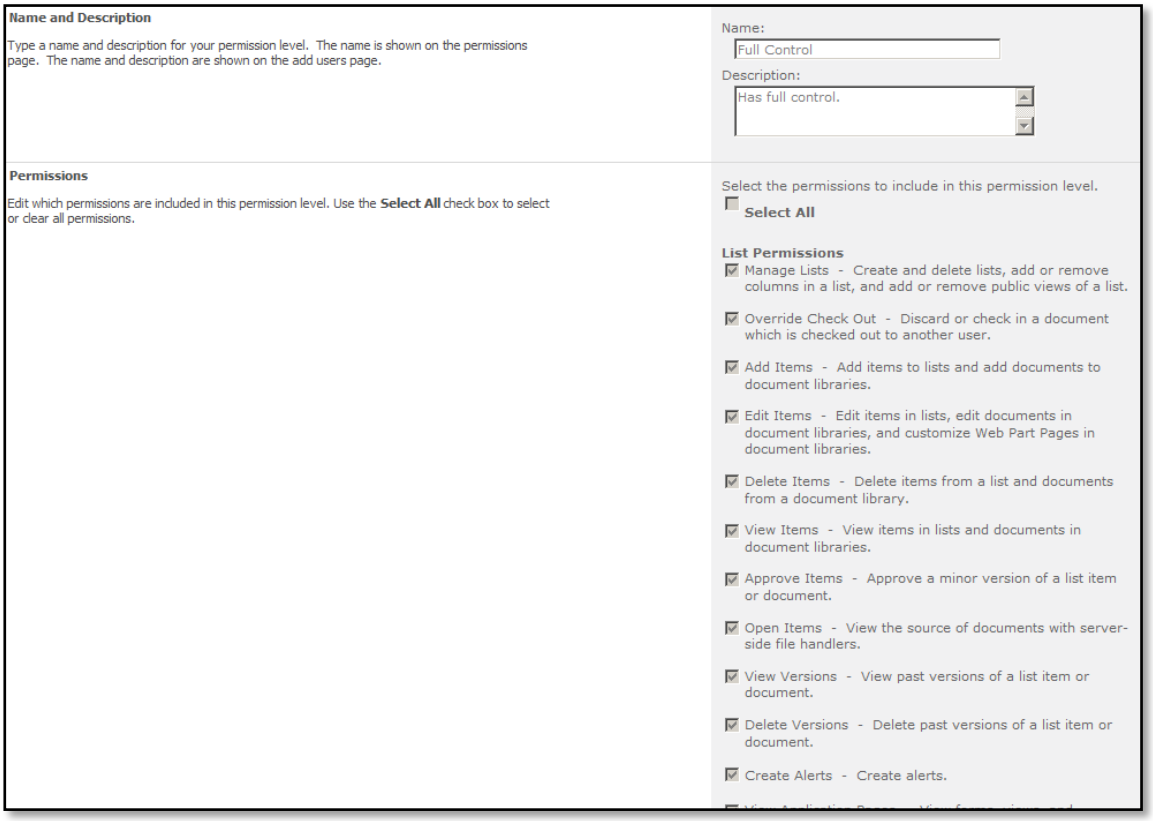
RSS-Feed



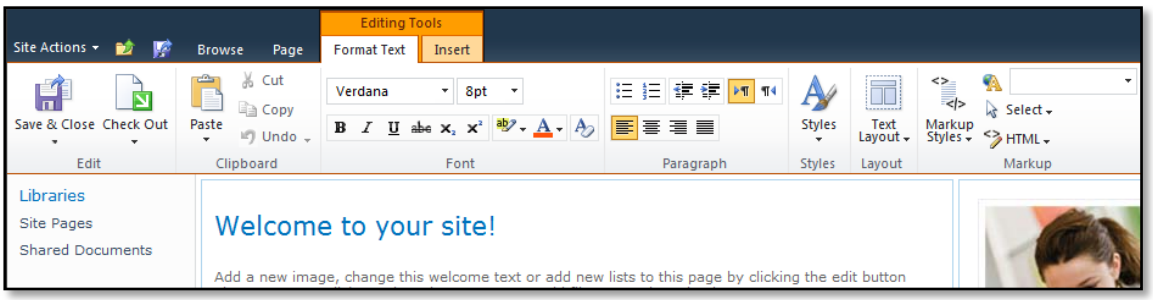
Kollaborationsbereich – Einstellungen



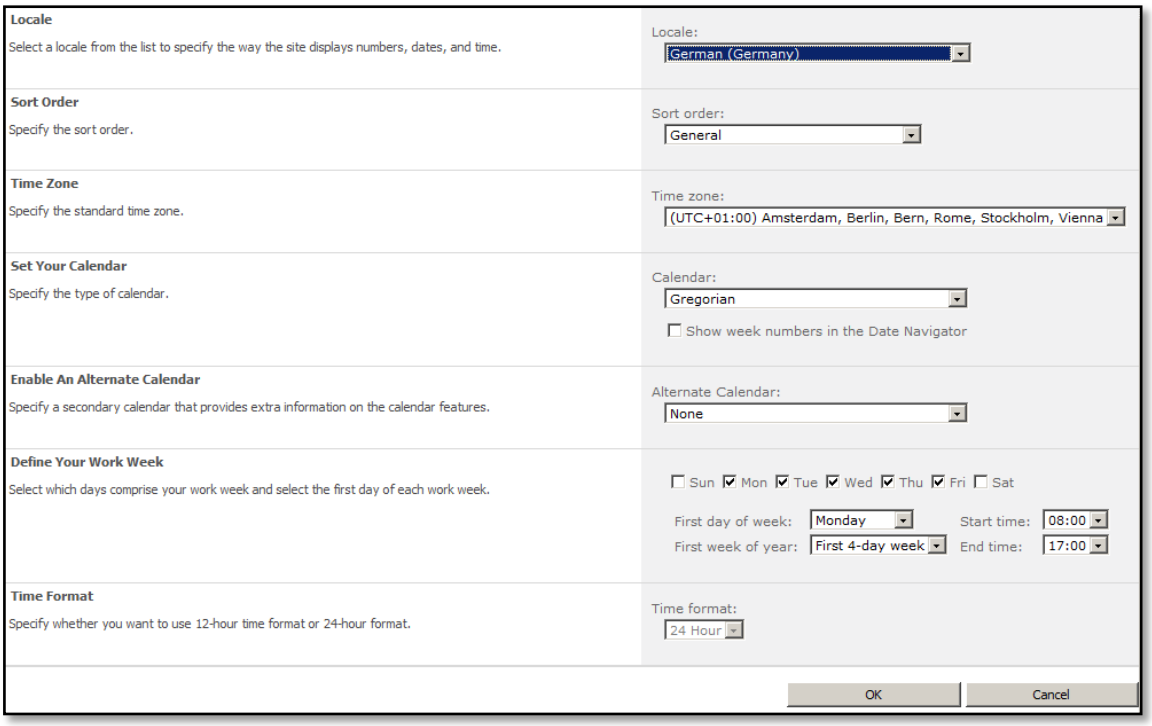
Kollaborationsbereich – Rechtevergabe



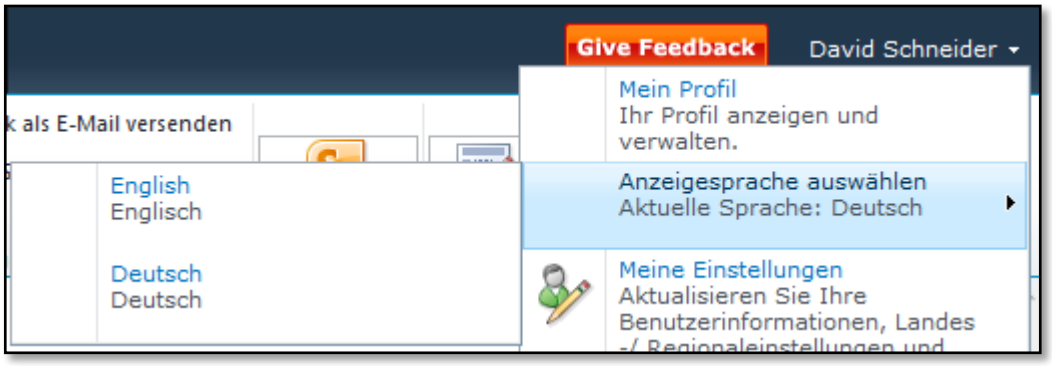
Kollaborationsbereich – Rechte-Level



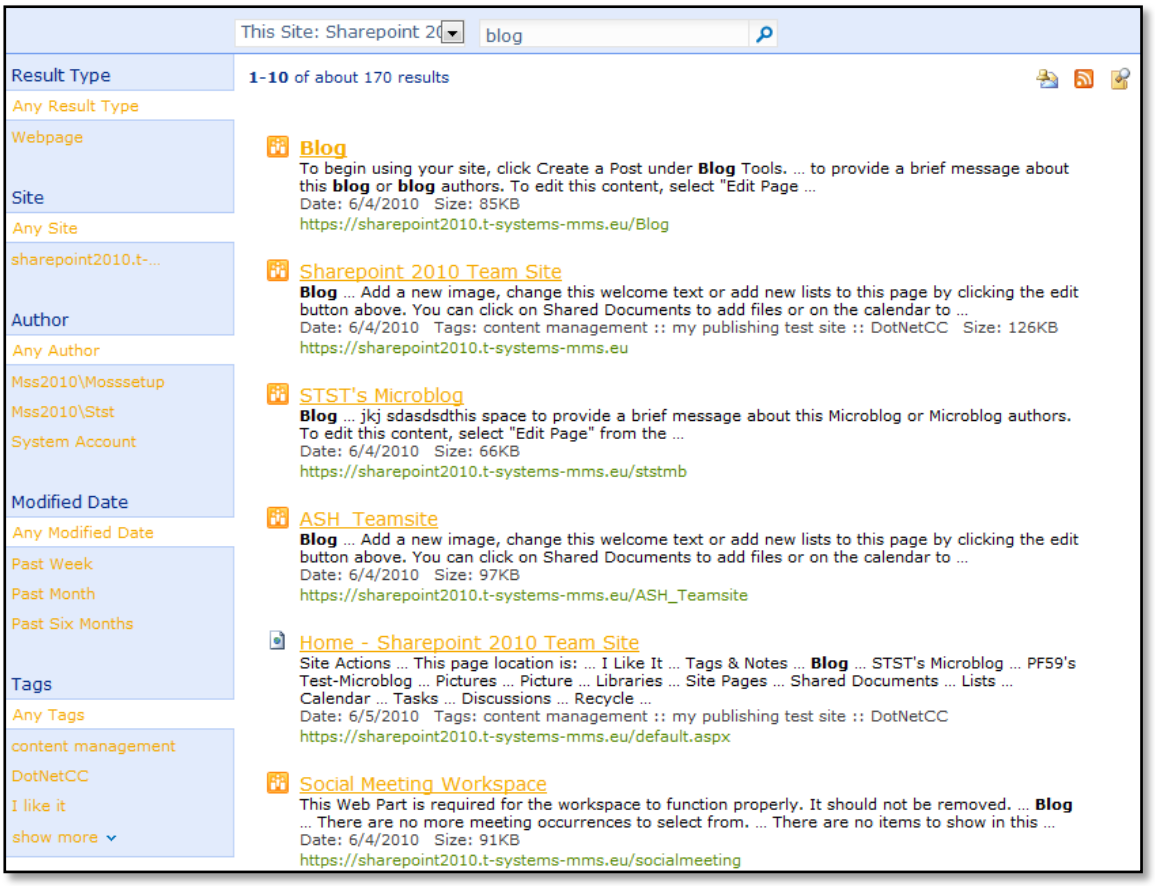
WYSIWYG-Editor



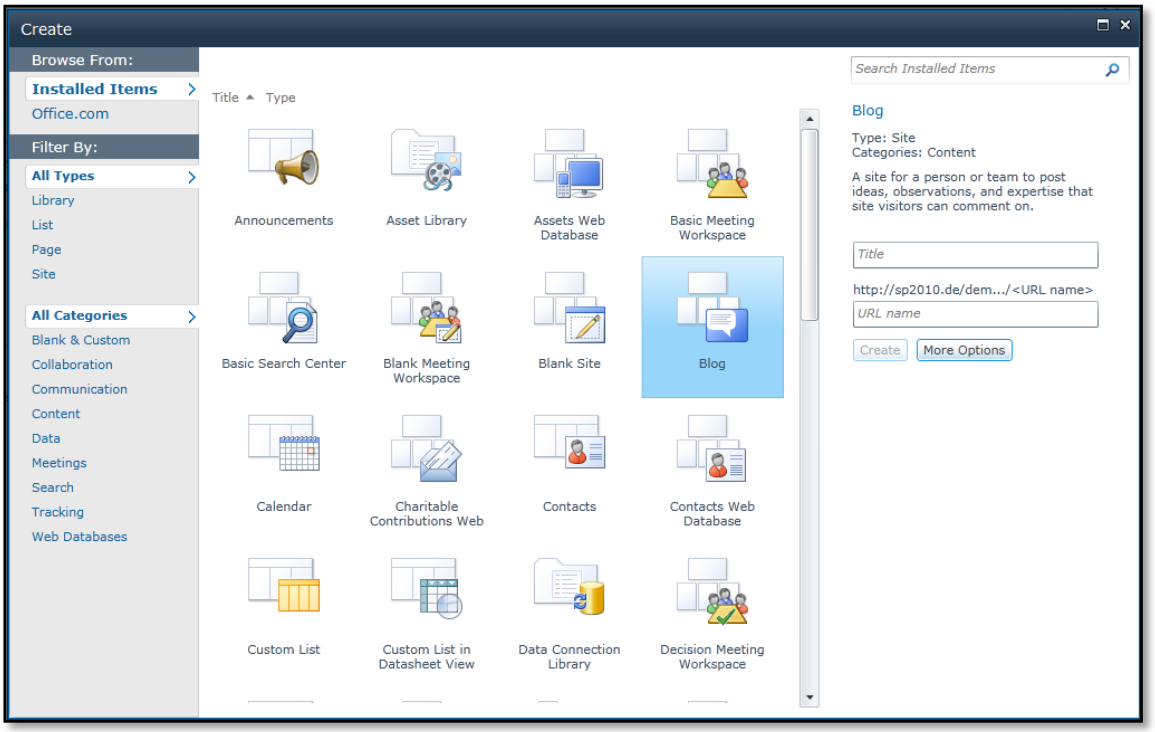
Regionale Einstellungen



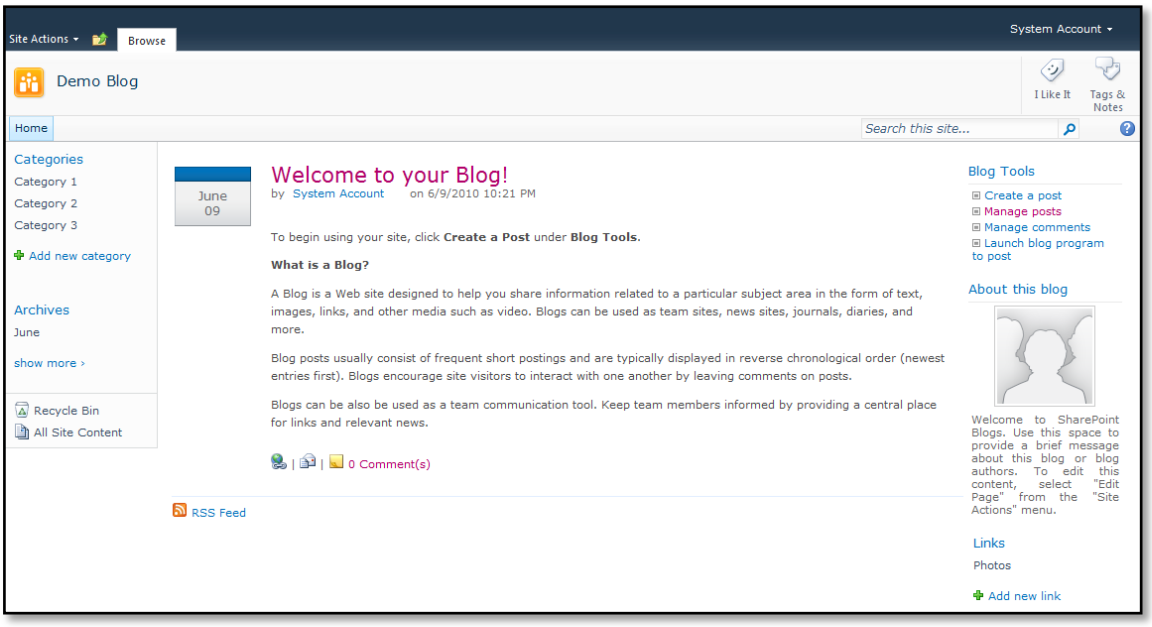
Auswahl Anzeigesprache



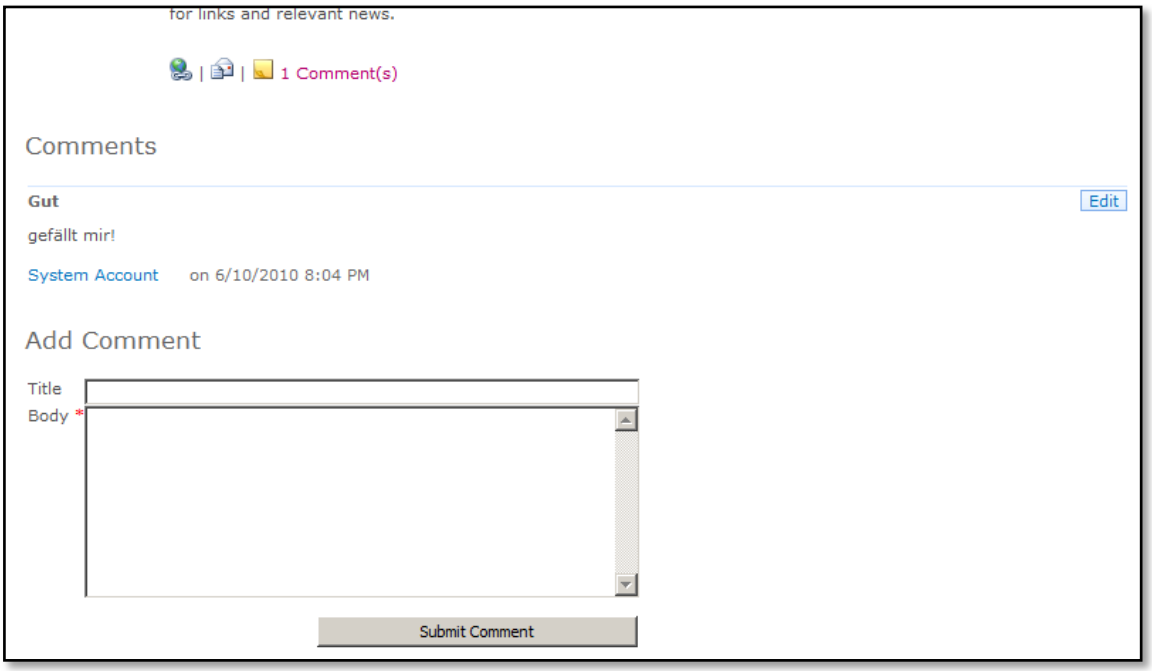
Globale Suche



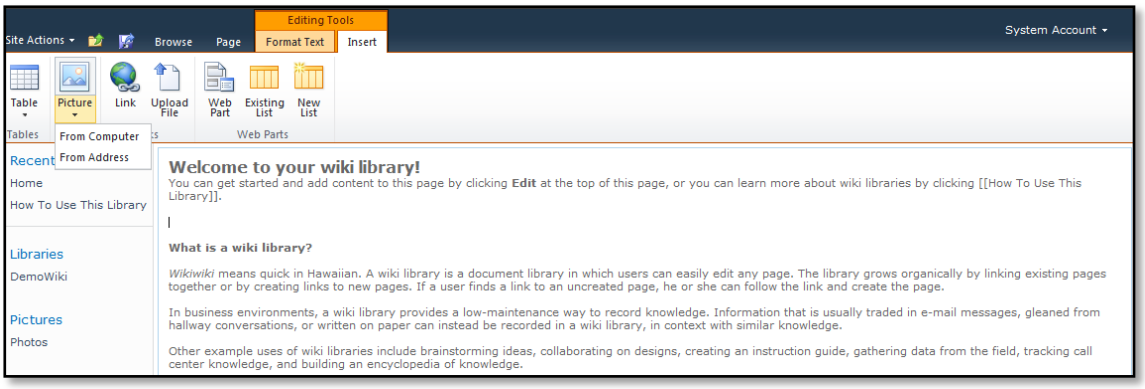
Erstellen Blog-Website



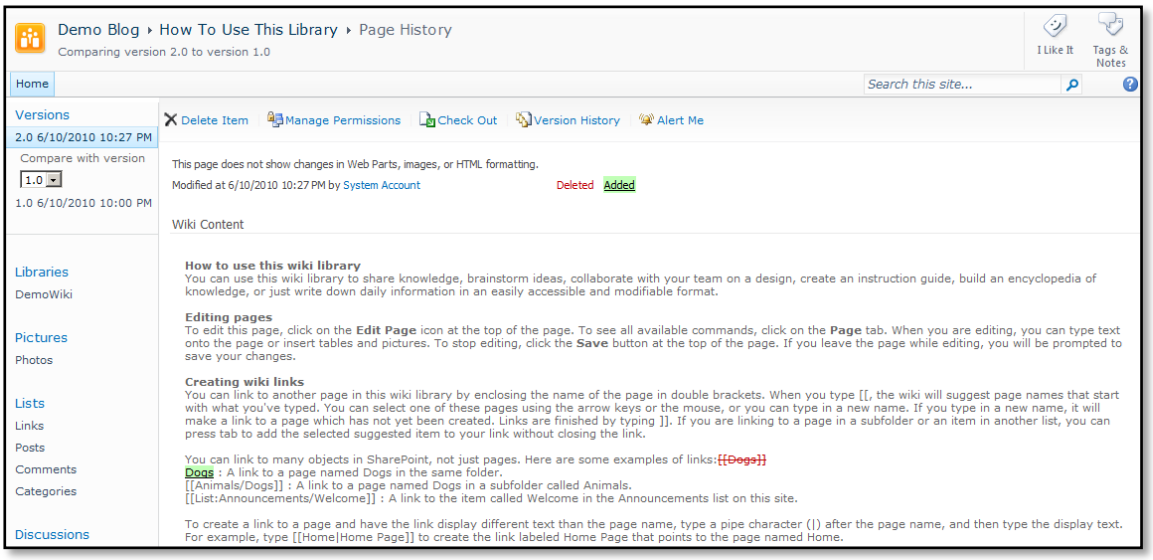
Blog



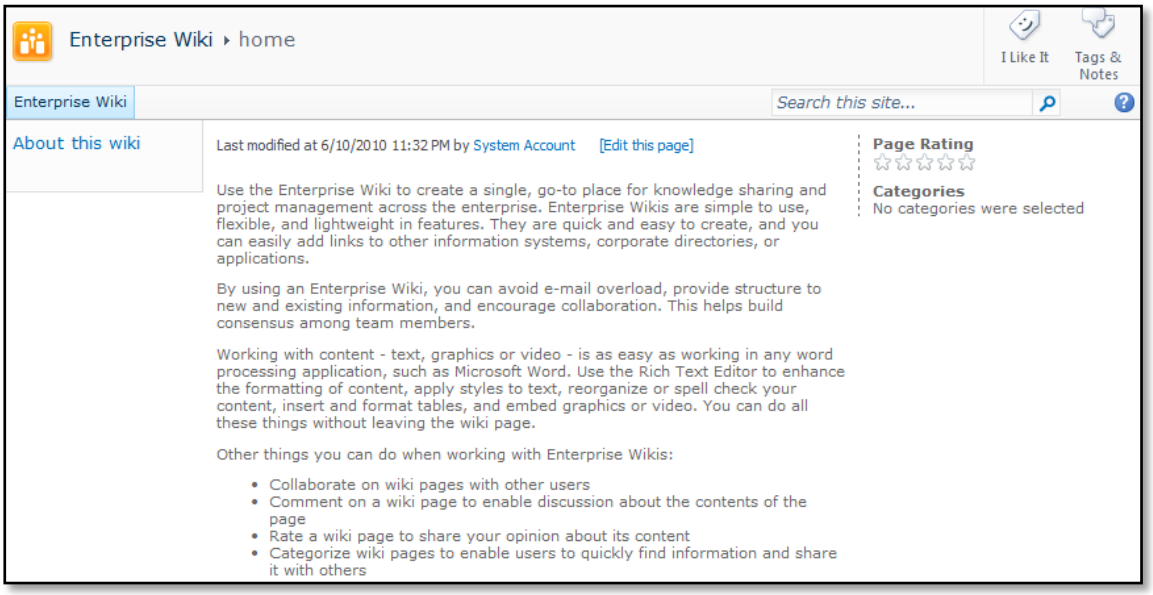
Blog – Kommentare



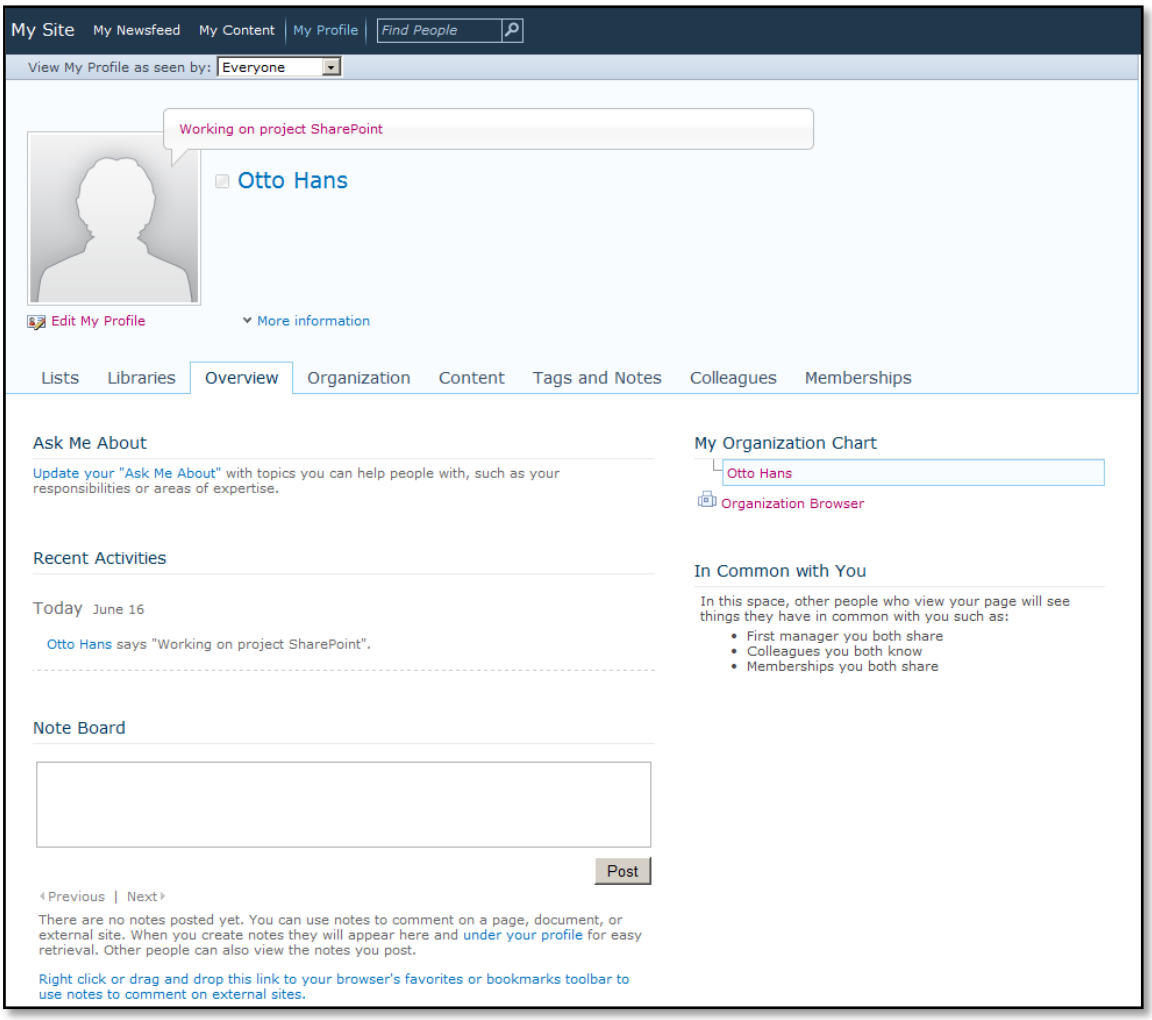
Wiki



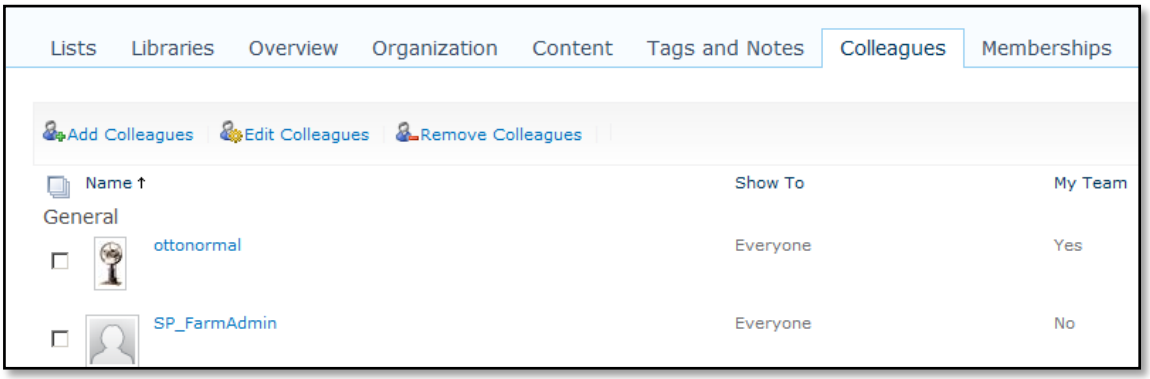
Wiki – Versionierung



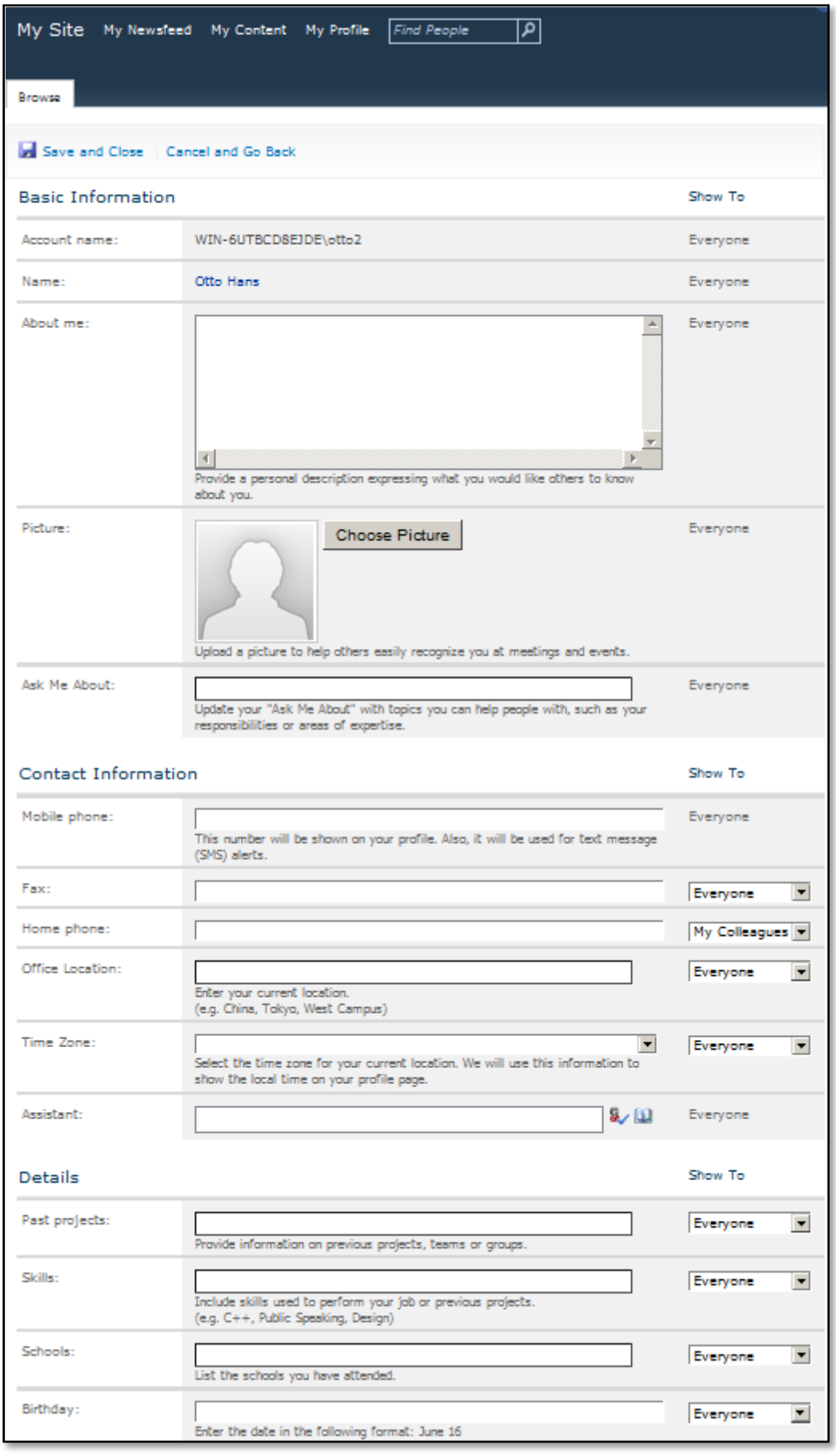
Enterprise-Wiki



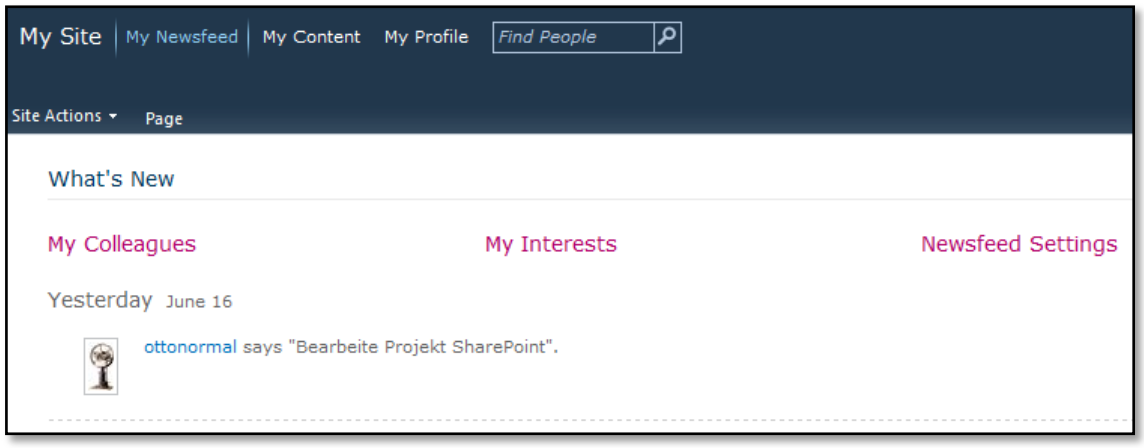
Social Networking-Services – Profil



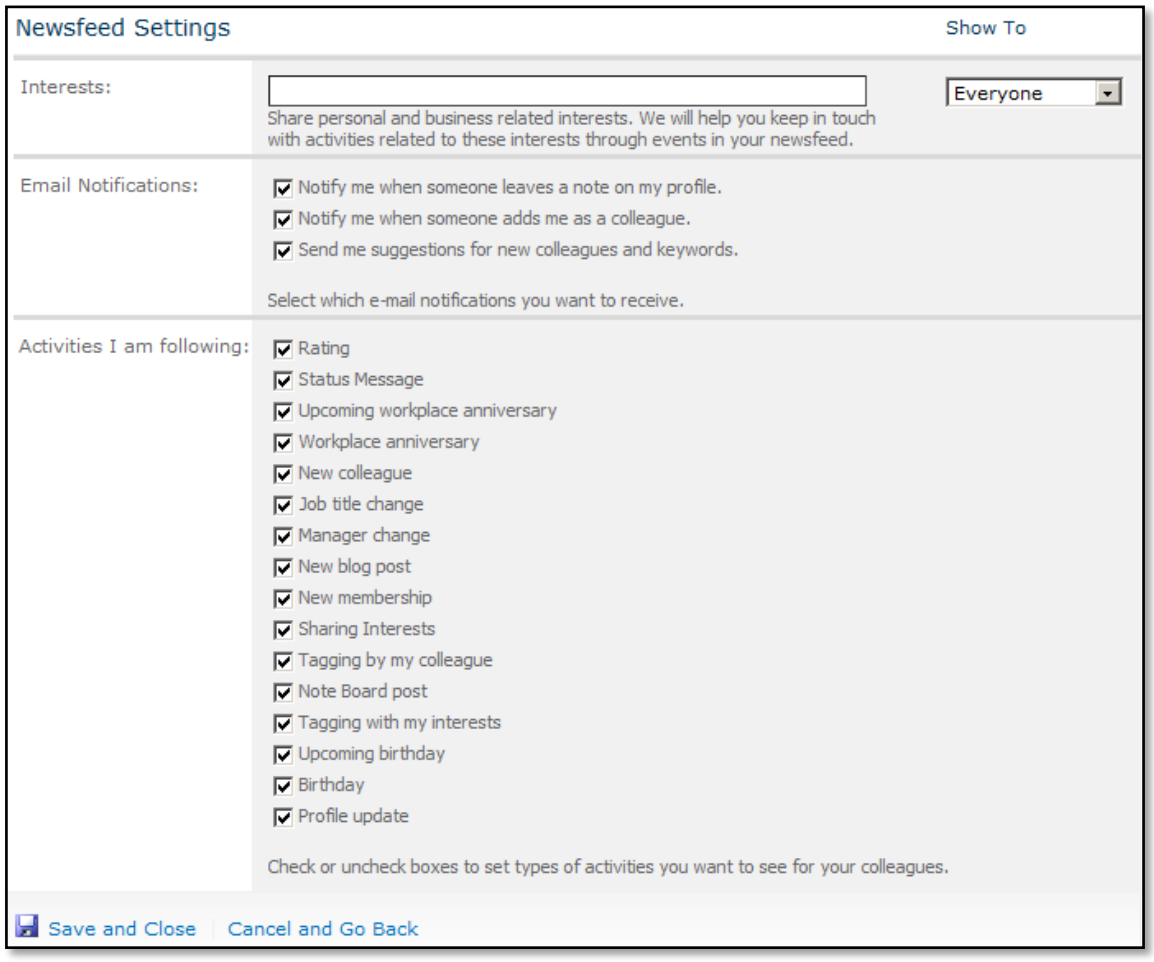
Social Networking-Services – Kollegenliste



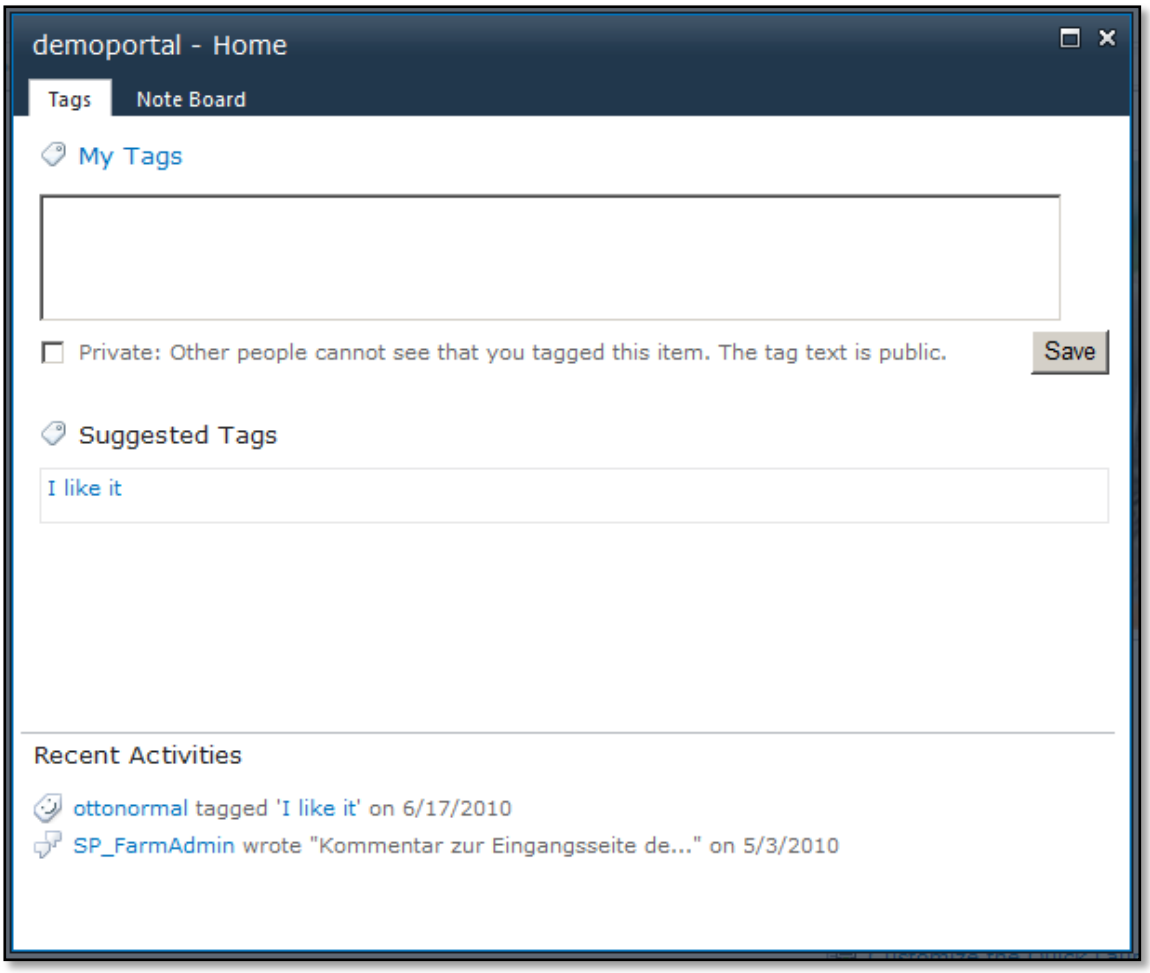
Social Networking-Services – Profil bearbeiten



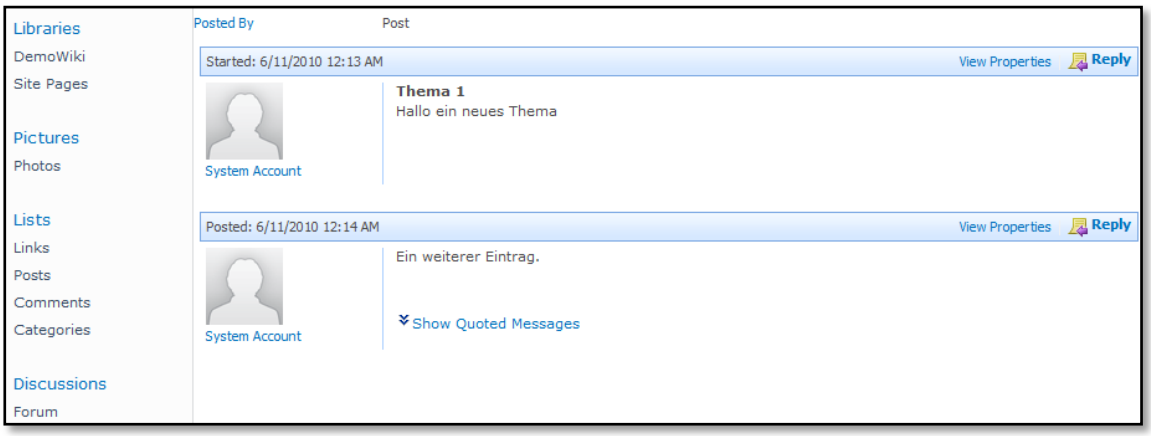
Activity Stream/My Newsfeed



Activity Stream/My Newsfeed – Einstellungen

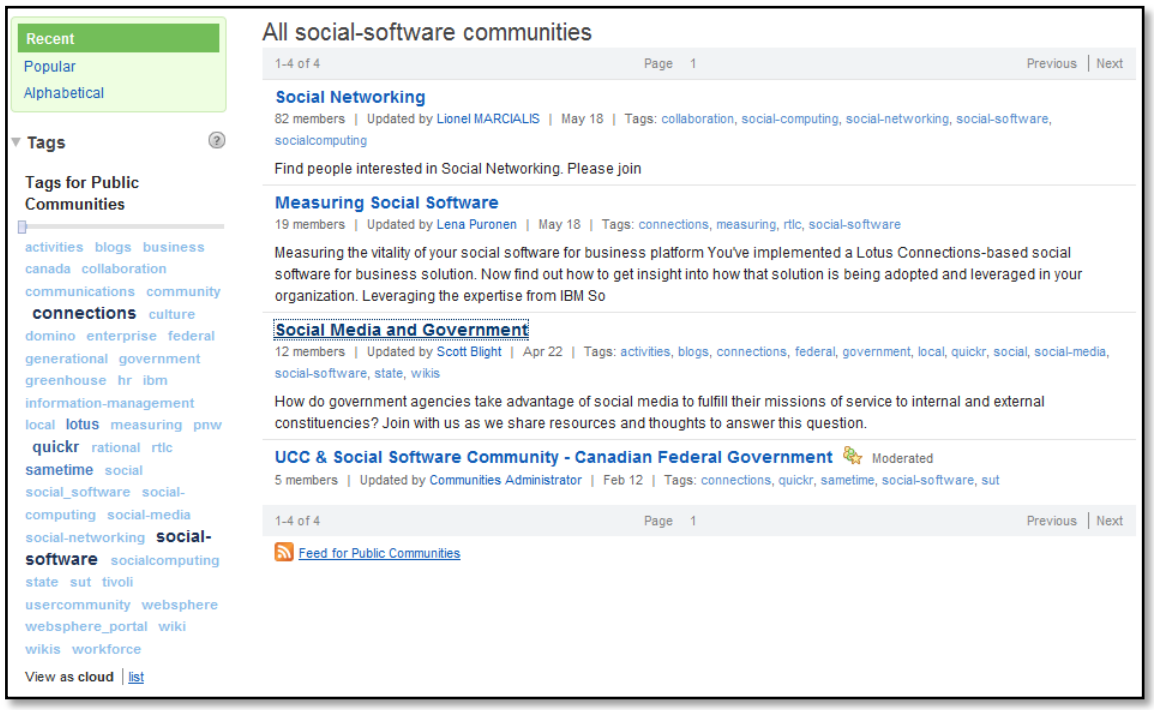


Social Tagging

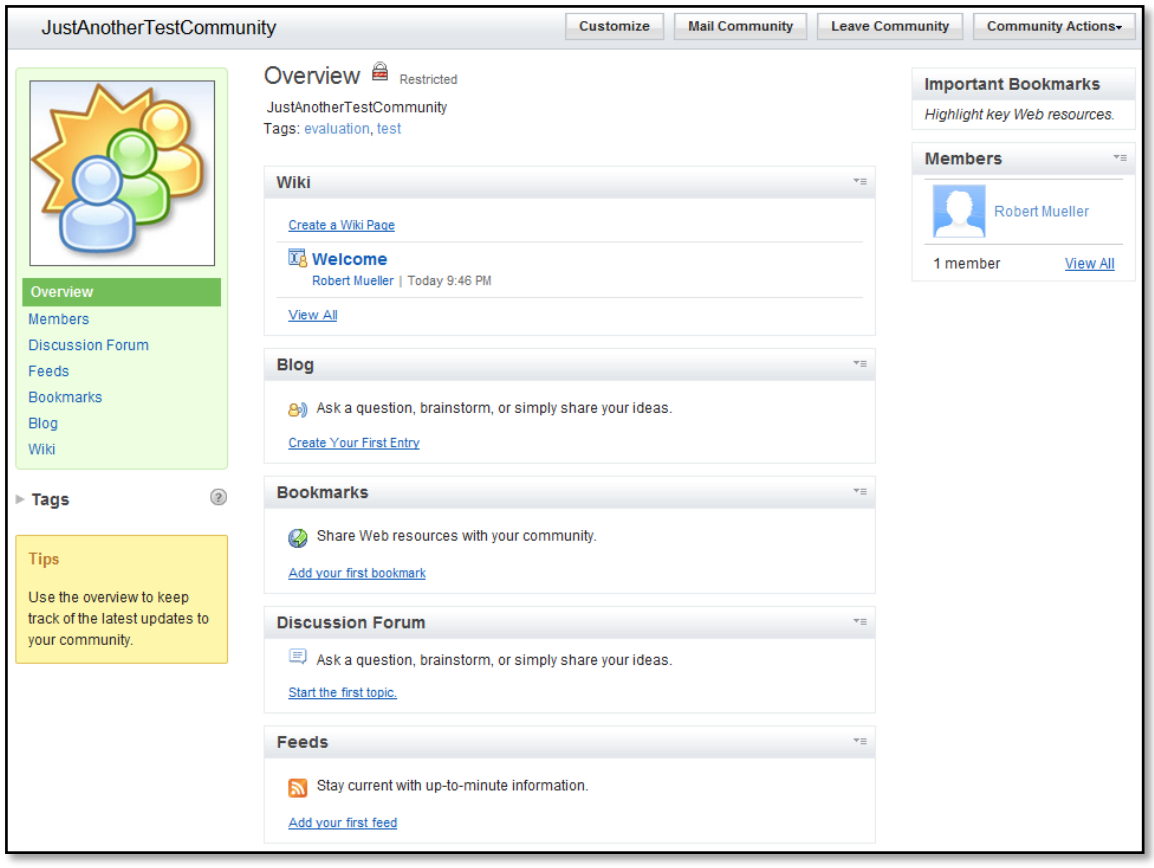


Forum

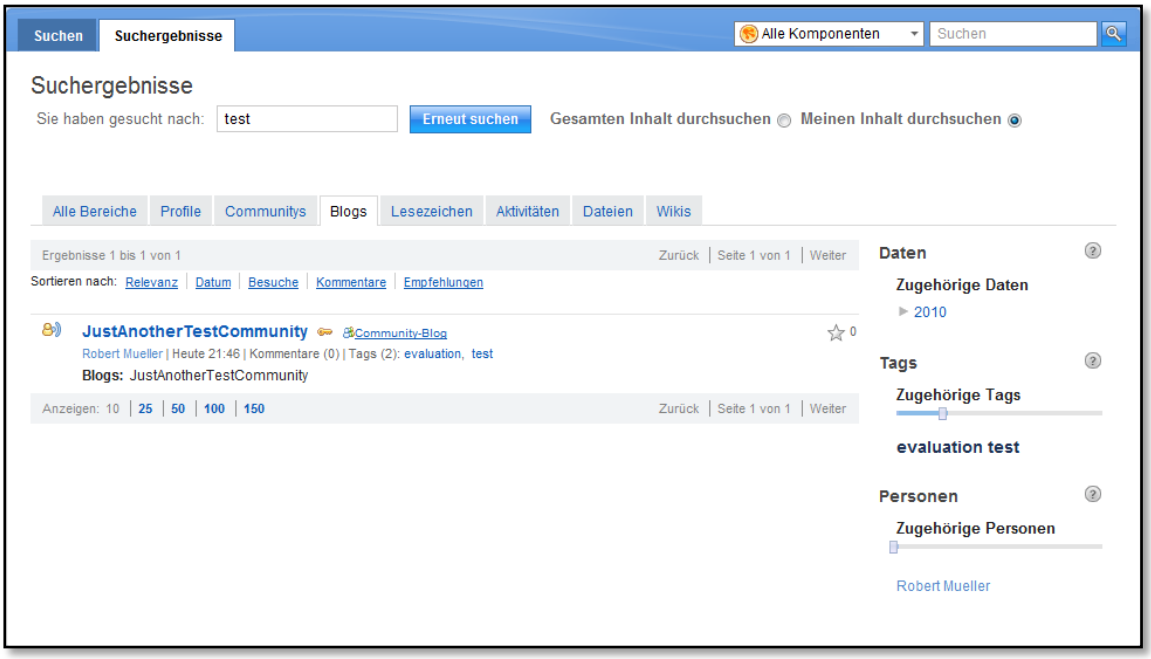
Anhang B – Bilder IBM Lotus Connections



Kollaborationsbereich – Communities-Liste

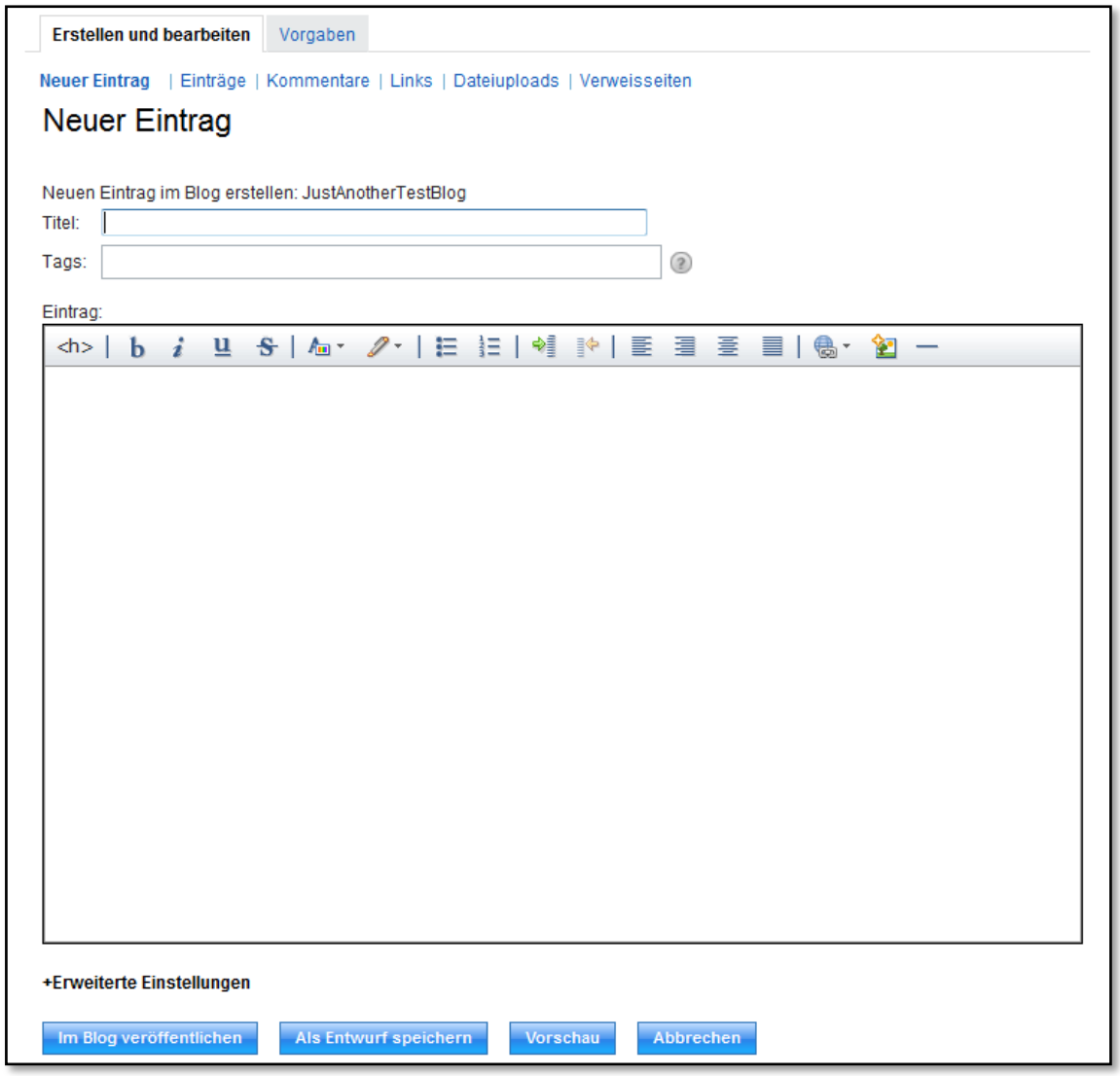


Kollaborationsbereich – Community



Globale Suche

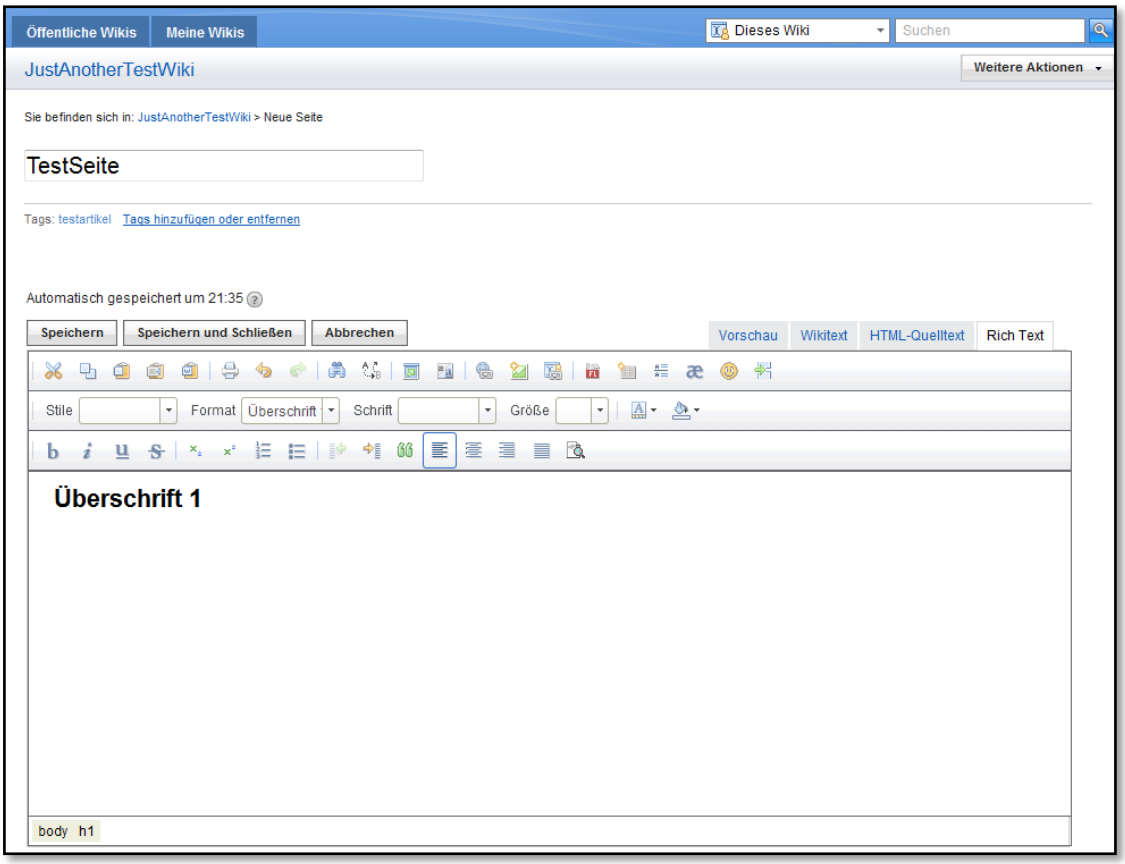
Blog



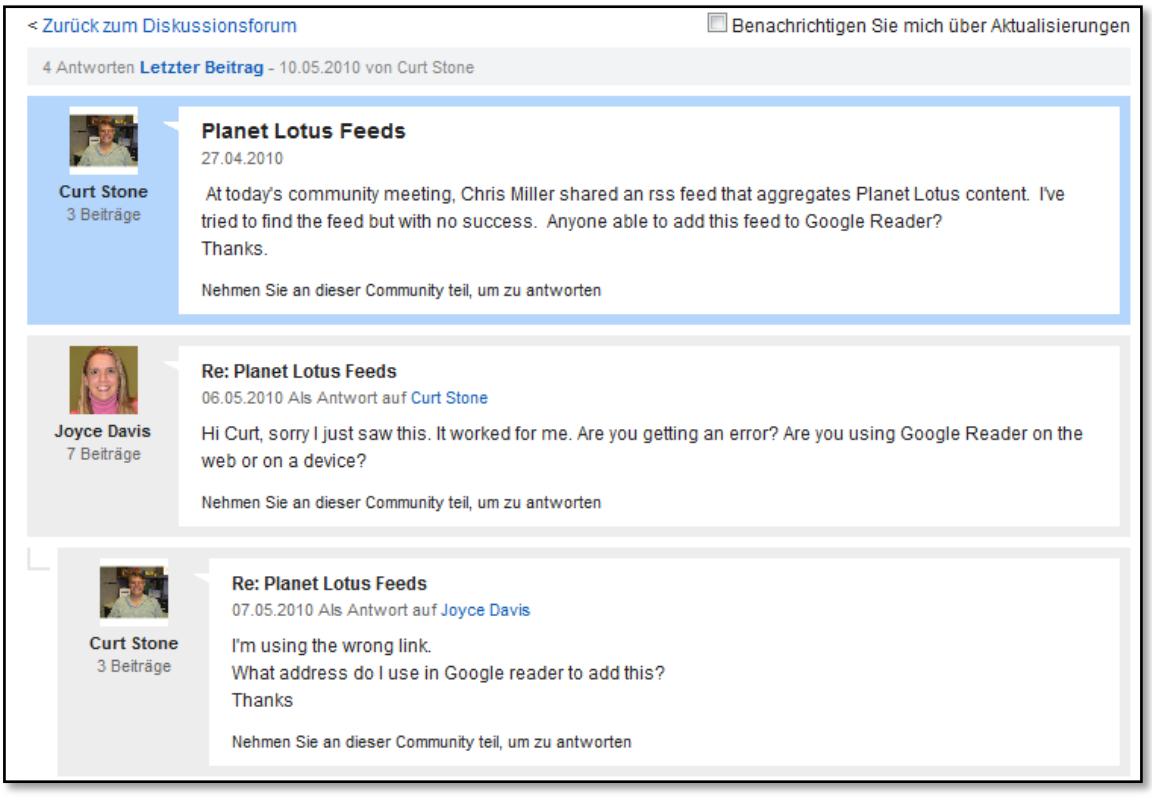
Blog – WYSIWYG-Editor


Status Updates
What's happening in your network right now?
Dan Misawa (no status message set)
I'm back from a great week at the show in LA. The event summary and show notes will be on Files shortly.
Post Status
Cancel
Show: Status Updates in My Network | My Status Updates
Dan Misawa Preparing for LA industry show Today 12:37 PM
(2 comments)
Frank Adams commented on 7/31/09 4:04 PM
Dan, let me know how that goes. I may want to replicate it for NY, next month.
Dan Misawa commented on 8/3/09 4:37 PM
Frank, I'm just back and have great ideas for your show in New York.
Write another comment...
Frank Adams finalizing my travel for next week Jul 31
(1 comment)

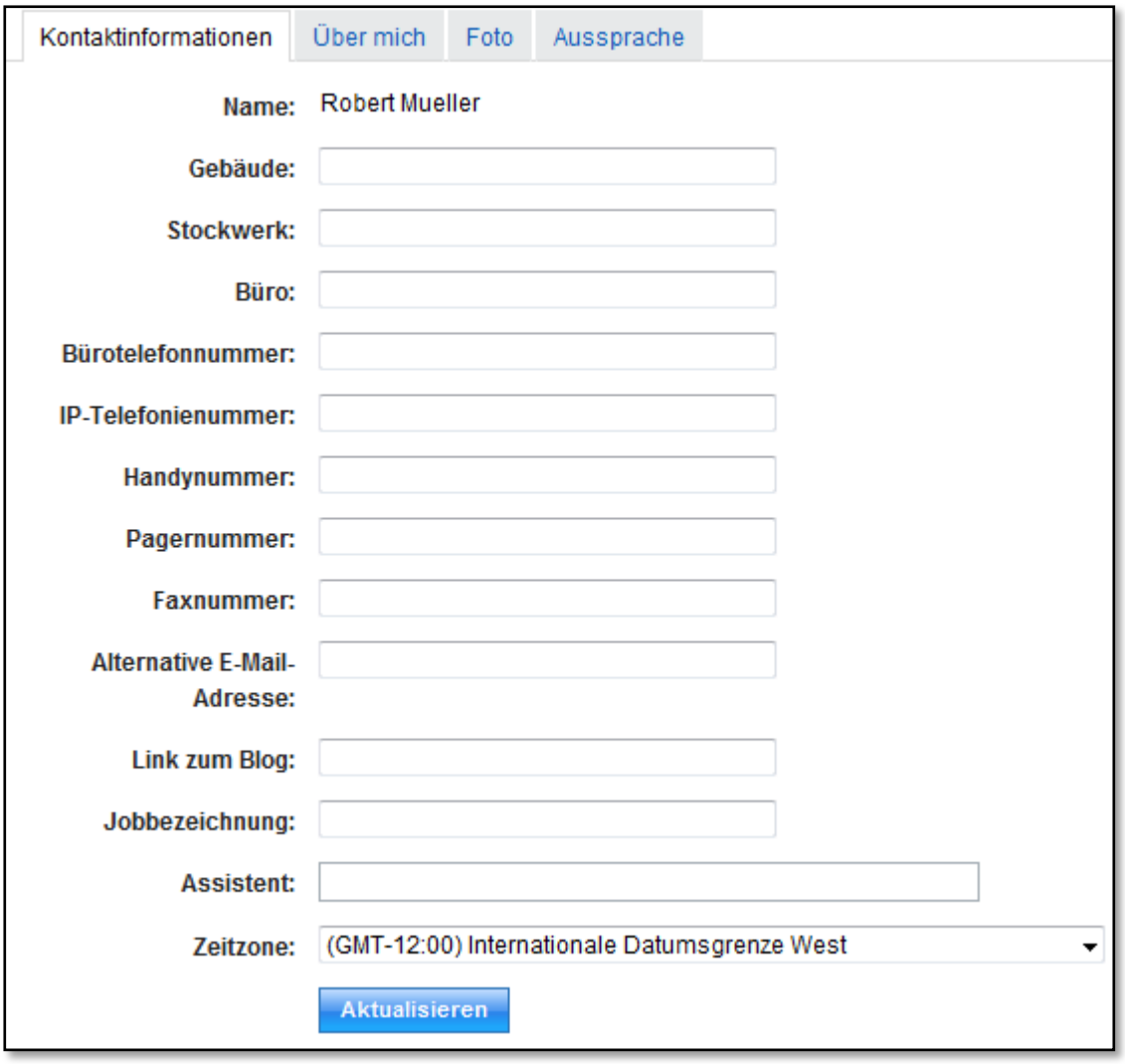
Microblog



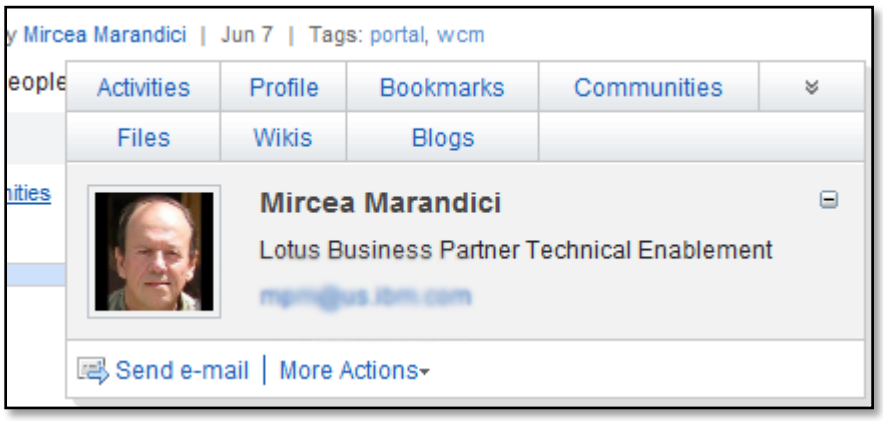
Wiki – WYSIWYG-Editor



Forum

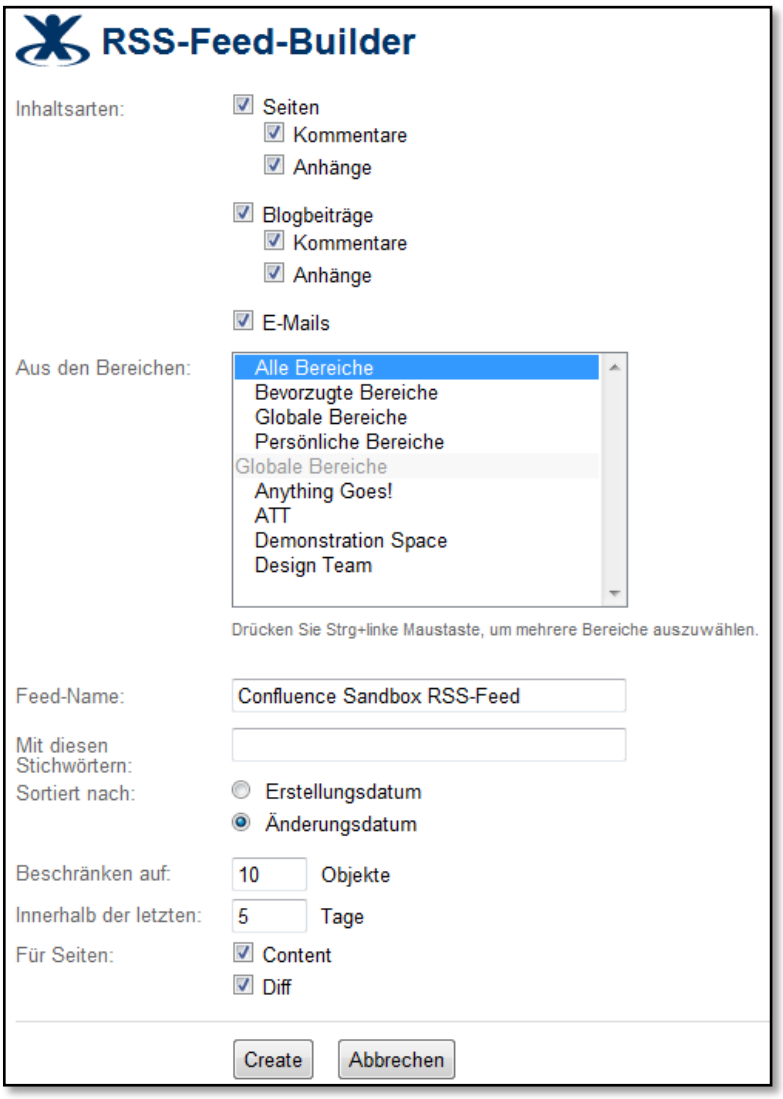


Social Networking-Services – Profil bearbeiten

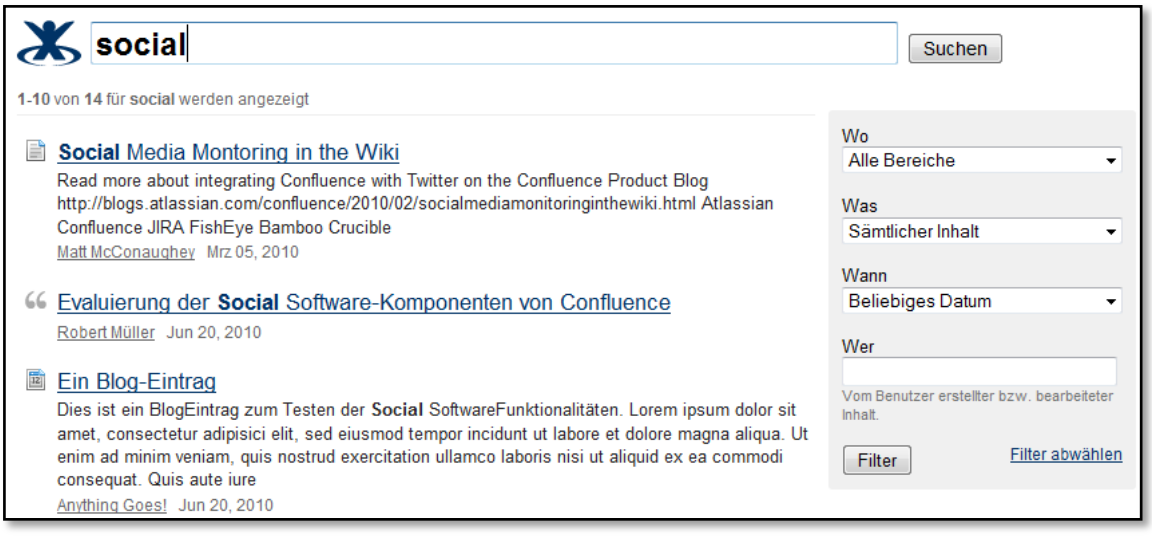


Social Networking-Services – Badge

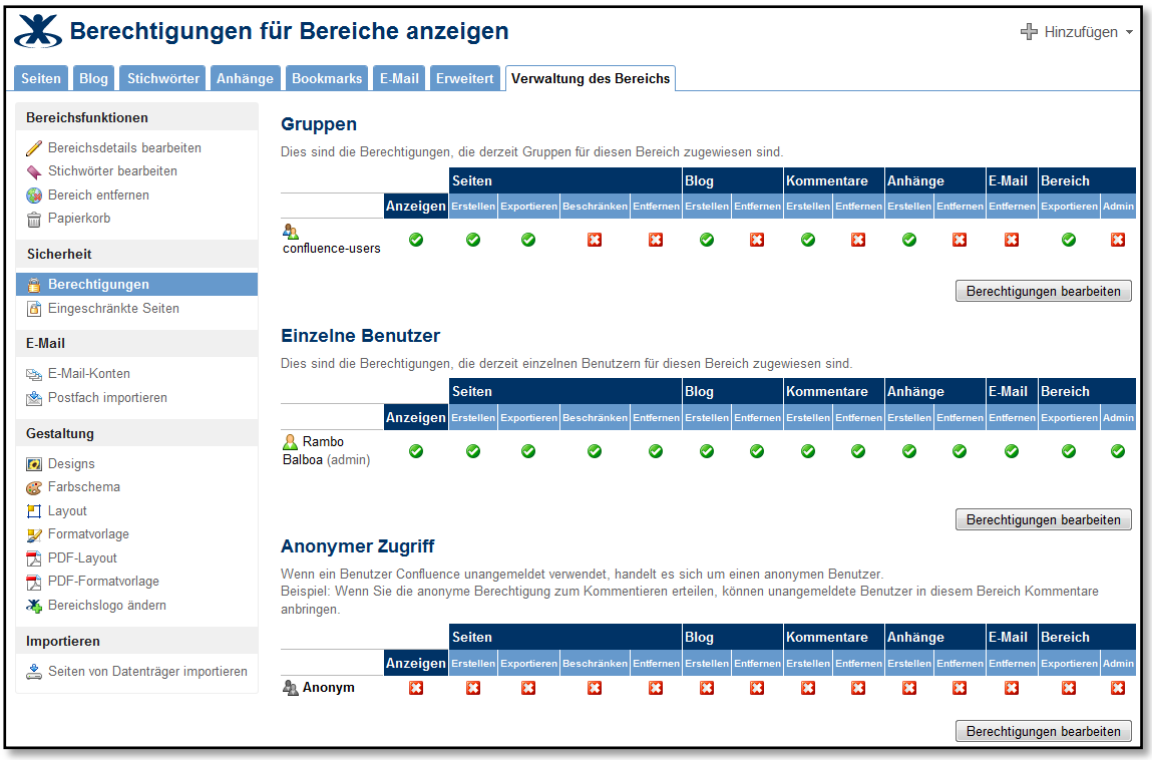
Anhang C – Bilder Atlassian Confluence



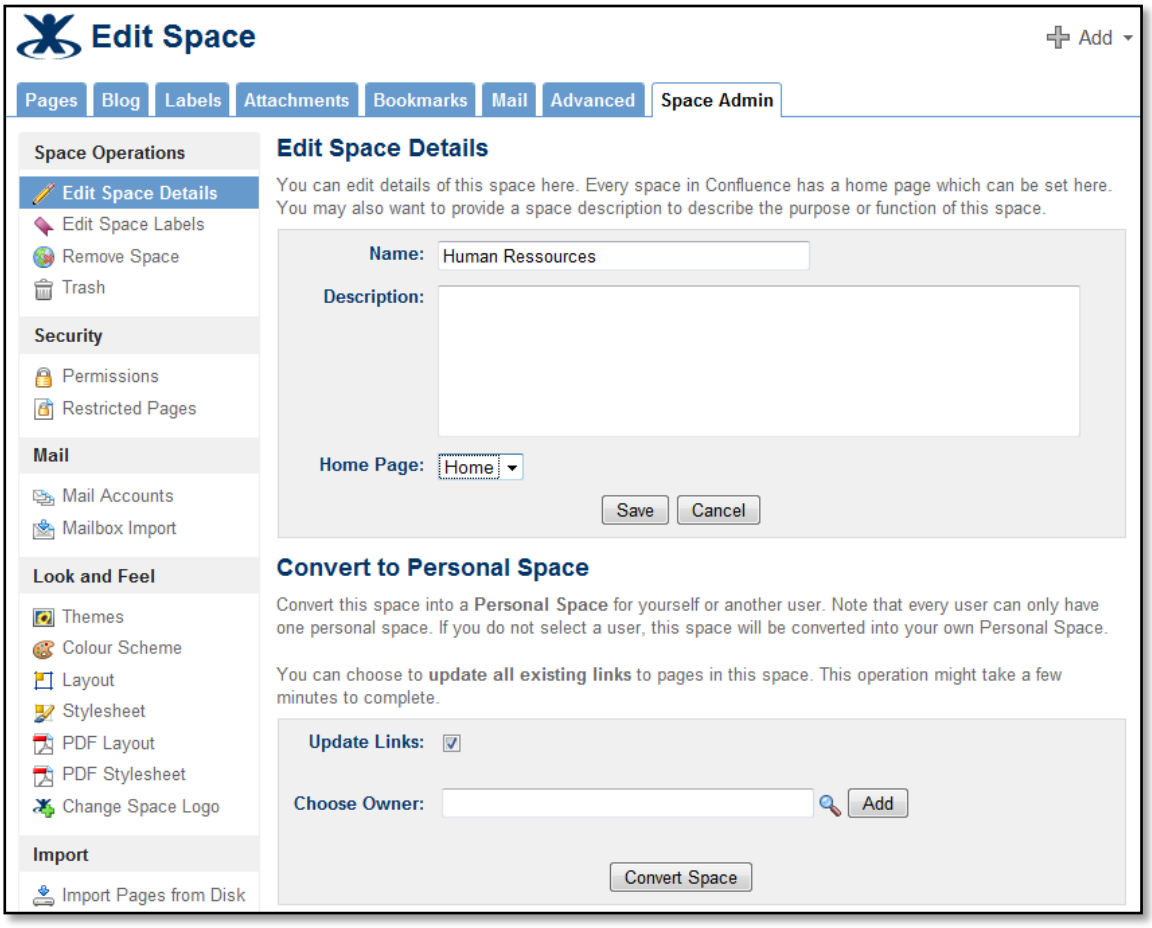
RSS-Feed-Builder



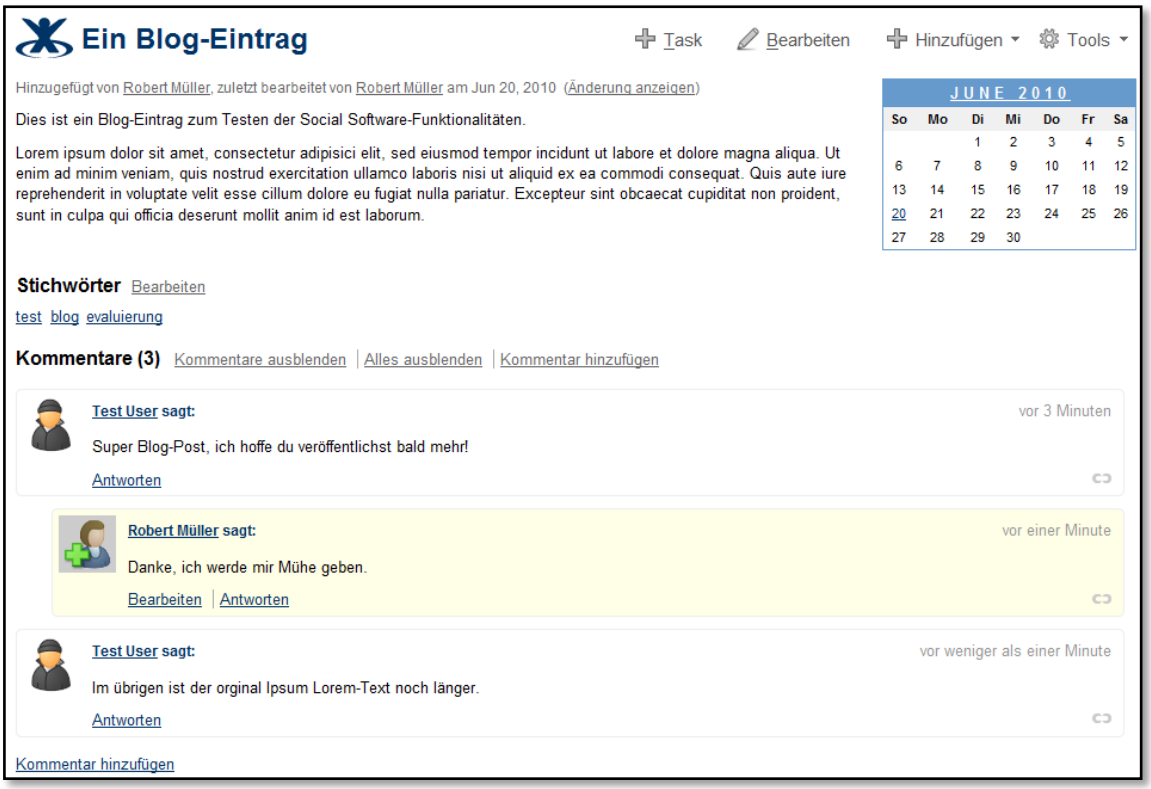
Globale Suche



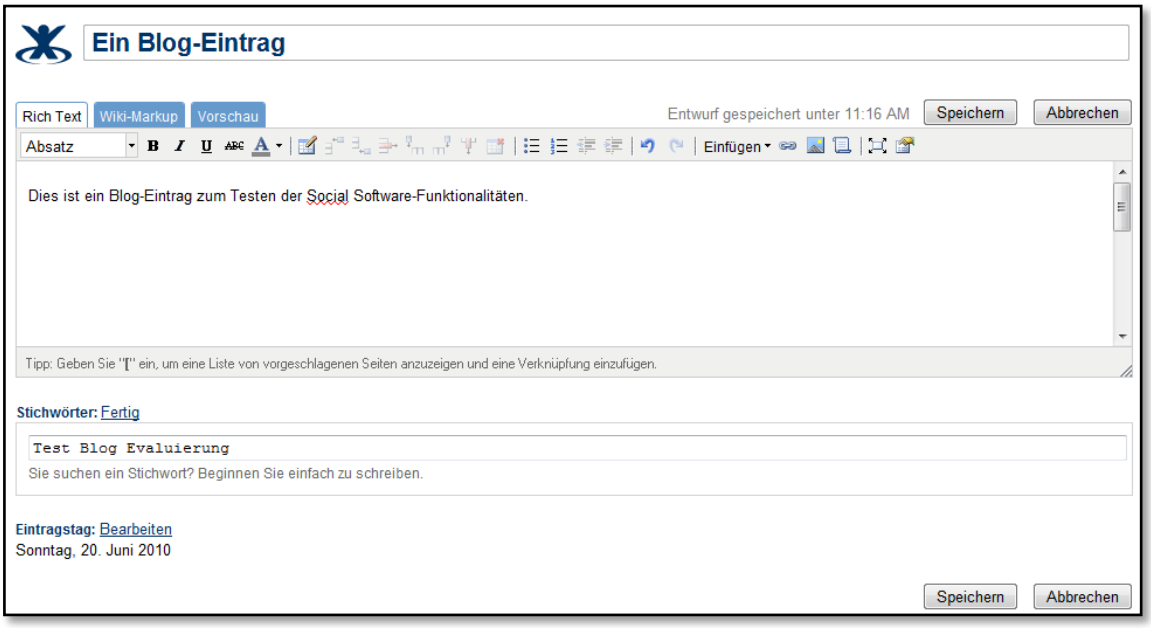
Kollaborationsbereich – Rechtevergabe



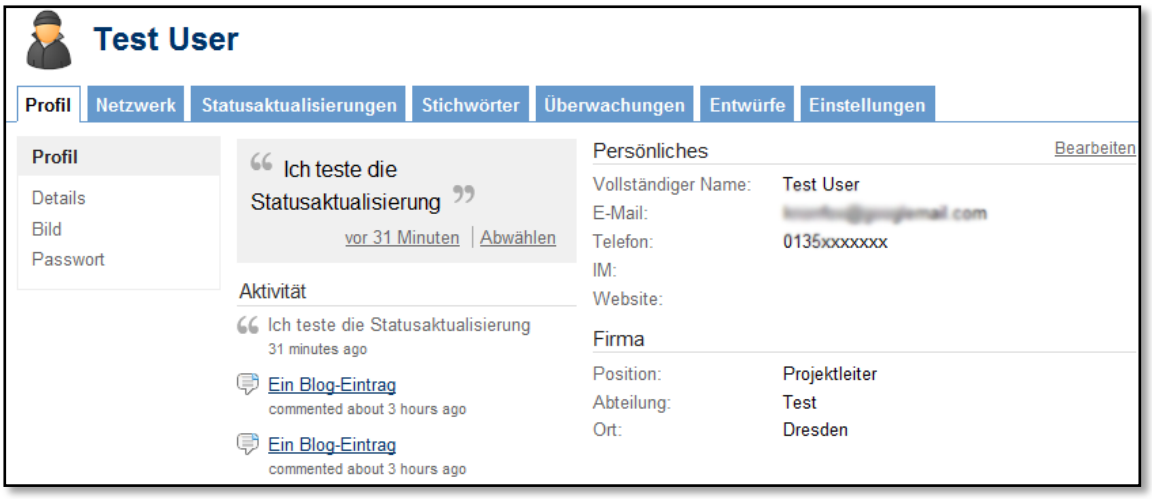
Kollaborationsbereich – Edit Space



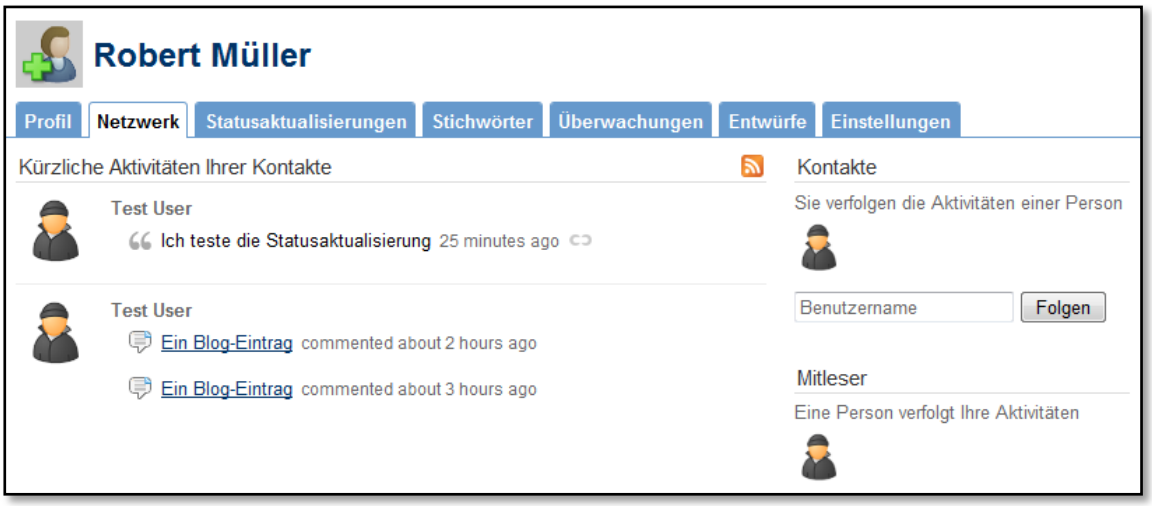
Blog



Blog – WYSIWYG-Editor

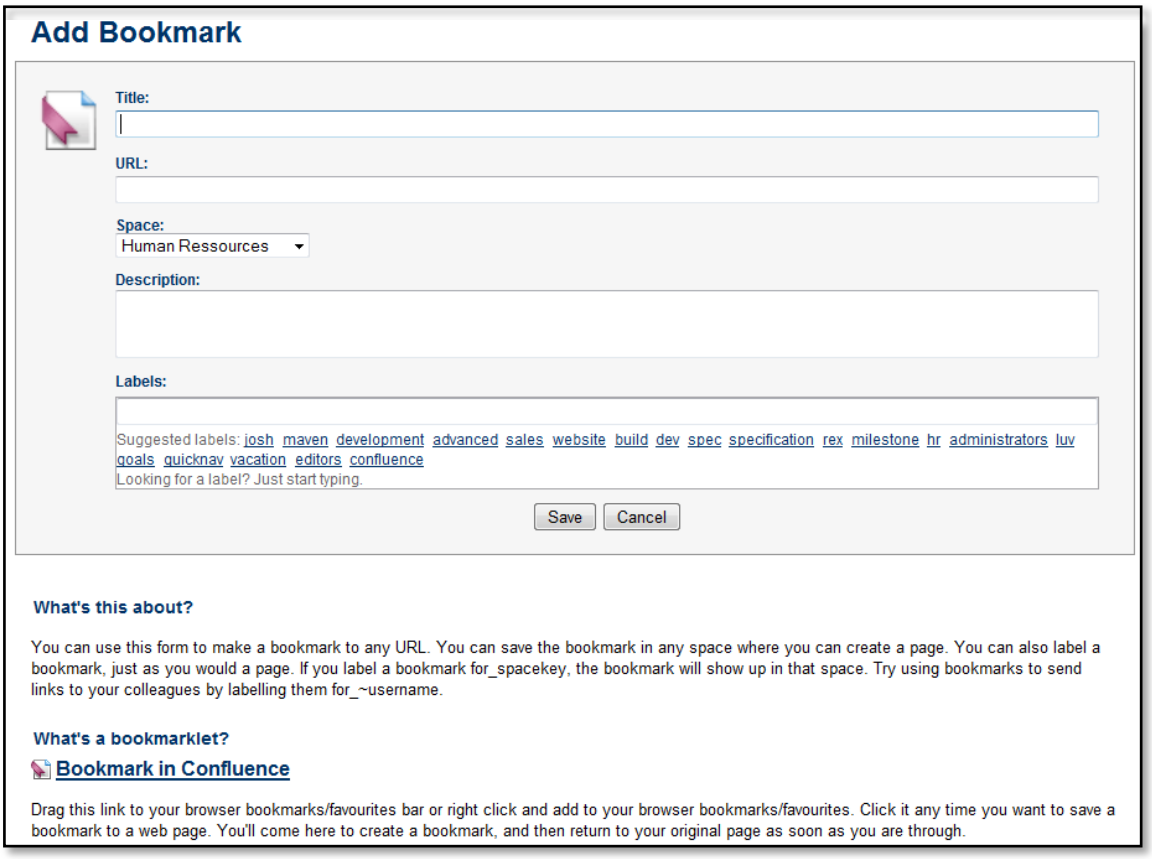


Profil



Profil – Activity Stream

Social Networking-Services – Badge



Social Bookmarking

Anhang D – Quellcode MicroBlogVisualWebPartUserControl.ascx.cs

```
1   using System;
2   using System.Web.UI;
3   using System.Web.UI.WebControls;
4   using System.Web.UI.WebControls.WebParts;
5   using Microsoft.SharePoint;
6   using Microsoft.Office.Server.UserProfiles;
7   using Microsoft.SharePoint.Utilities;
8   using System.Data;
9   using System.Linq;
10  using System.Text.RegularExpressions;
11  using System.Collections.Generic;
12
13  namespace MicroblogSolution.MicroBlogVisualWebPart
14  {
15   public partial class MicroBlogVisualWebPartUserControl : UserControl
16   {
17
18    int tweetLengthLimit = 160; //tweetLengthLimit, bei über 160 auch Field (MaxLength)
19     in Schema.xml (MicroBlogList) anpassen
20    public int tweetEntries; //Anzahl der angezeigten Einträge
21    public int quantityTags; //Anzahl der angezeigten Tags
22    public int weeksAgo; //Anzahl der vergangenen Wochen, die bei der TagCloud-
23     Erstellung einbezogen werden sollen
24    MicroBlogVisualWebPart ms;
25
26    protected void Page_Init(object sender, EventArgs e)
27    {
28     //Custom Webpart Properties in UserControl übergeben
29     ms = (MicroBlogVisualWebPart)Parent;
30     tweetEntries = ms.TweetEntries;
31     quantityTags = ms.QuantityTags;
32     weeksAgo = ms.WeeksAgo;
33
34     if (!IsPostBack)
35     {
36      //TweeTable bei ersten Seitenaufruf erzeugen
37      getMicroblogTweets();
38     }
39     setInputLimit(tweetLengthLimit);
40     generateTagCloud();
41
42     searchTextBox.Attributes.Add("onfocus", "javascript:SearchText(this);");
43     searchTextBox.Attributes.Add("onBlur", "javascript:SearchTextNew(this);");
44    }
45
46    private void generateTagCloud()
47    {
48    SPSite currentSite = SPContext.Current.Site;
49    String siteUrl = currentSite.Url;
50    String currentWebUrl = SPContext.Current.Web.Url;
51    SPServiceContext context = SPServiceContext.GetContext(currentSite);
52    UserProfileManager myUPManager = new UserProfileManager(context);
53    String loginName = SPContext.Current.Web.CurrentUser.LoginName;
54
55    Dictionary<string, double> tagDictBlog = new Dictionary<string,
56     double>(StringComparer.OrdinalIgnoreCase);
57    Dictionary<string, double> tagDictAll = new Dictionary<string,
58     double>(StringComparer.OrdinalIgnoreCase);
59    Dictionary<string, double> tagDictUser = new Dictionary<string,
60     double>(StringComparer.OrdinalIgnoreCase);
61
```

```
62      double daysAgo = (weeksAgo * -7);
63      SPQuery query = new SPQuery();
64      query.ExpandUserField = false;
65      query.Query = "<Where><And><IsNotNull><FieldRef Name='HashTags' /></IsNotNull>" +
66        "<Gt><FieldRef Name='Created' /><Value Type='DateTime'><Today OffsetDays=" + day-
67        sAgo.ToString() + "/></Value>" + "</Gt></And></Where>";
68
69      UserProfile uProfileCurrent = ProfileLoader.GetProfileLoader().GetUserProfile();
70      String currentUserAccountName = uProfileCurrent["AccountName"].ToString();
71      String userTag = "@" + uProfileCurrent["PreferredName"].ToString();
72
73      //Auflistung aller Microbloglisten (ElevatedPrivileges notwendig wegen AllWebs)
74      SPSecurity.RunWithElevatedPrivileges(delegate()
75      {
76        SPSite cSite = new SPSite(siteUrl);
77        //Die Listen für Alle Seiten in tweetTable mergen
78        foreach (SPWeb web in cSite.AllWebs)
79        {
80          //Überprüfen, ob der Nutzer auf die Liste zugreifen darf
81          if ((checkListExistence(web, "Microblog List") &&
82            (web.DoesUserHavePermissions(loginName, SPBasePermissions.ViewPages))))
83          {
84            //DataTable mit Listeinträgen füllen, die HashTags beinhalten
85            DataTable tempTable = new DataTable();
86            tempTable = web.Lists["Microblog List"].GetItems(query).GetDataTable();
87            if ((tempTable != null) && (tempTable.Rows.Count > 0))
88            {
89              for (int i = 0; i < tempTable.Rows.Count; i++)
90              {
91                String[] hashTags = tempTable.Rows[i]["HashTags"].ToString().Split(new
92                  Char[] { ' ' });
93
94                //Überprüfen, ob tagDictUser befüllt werden muss
95                bool userScope = false;
96                if (myUPManager.UserExists(tempTable.Rows[i]["Author"].ToString()))
97                {
98                  if ((tempTable.Rows[i]["Tweet"].ToString().IndexOf(userTag, 0, StringCom-
99                    parison.OrdinalIgnoreCase) >= 0) || (currentUserAccountName == tempTa-
100                   ble.Rows[i]["Author"].ToString()))
101                 {
102                   userScope = true;
103                 }
104               }
105               //Überprüfen, ob tagDictBlog befüllt werden muss
106               bool blogScope = false;
107               if (currentWebUrl == web.Url)
108               {
109                 blogScope = true;
110               }
111               for (int hashTagIndex = 0; hashTagIndex < hashTags.Length; hashTagIndex++)
112               {
113                 //Dictionary für All Microblogs Scope füllen
114                 if (tagDictAll.ContainsKey(hashTags[hashTagIndex]))
115                 {
116                   tagDictAll[hashTags[hashTagIndex]] = tagDictAll[hashTags[hashTagIndex]] +
117                     1;
118                 }
119                 else
120                 {
121                   tagDictAll.Add(hashTags[hashTagIndex], 1);
122                 }
123                 //Dictionary für User Scope füllen
124                 if (userScope == true)
125                 {
126                   if (tagDictUser.ContainsKey(hashTags[hashTagIndex]))
127                   {
```

```
128                 tagDictUser[hashTags[hashTagIndex]] = tagDictUs-
129                   er[hashTags[hashTagIndex]] + 1;
130                }
131                else
132                {
133                 tagDictUser.Add(hashTags[hashTagIndex], 1);
134                }
135               }
136               //Dictionary für Microblog Scope füllen
137               if (blogScope == true)
138               {
139                if (tagDictBlog.ContainsKey(hashTags[hashTagIndex]))
140                {
141                 tagDictBlog[hashTags[hashTagIndex]] = tag-
142                   DictBlog[hashTags[hashTagIndex]] + 1;
143                }
144                else
145                {
146                 tagDictBlog.Add(hashTags[hashTagIndex], 1);
147                }
148               }
149              }
150             }
151            }
152           }
153          }
154         }
155         );
156         if (tagDictBlog.Count > 0)
157         {
158         DataTable tagCloudDataTableBlog = new DataTable();
159          tagCloudDataTableBlog.Columns.Add("HashTagName");
160          tagCloudDataTableBlog.Columns.Add("FontSize", typeof(FontUnit));
161
162          FontUnit[] fontSize = { FontUnit.Small, FontUnit.Medium, FontUnit.Large, FontU-
163           nit.XLarge };
164
165          //Tags, nach Aufkommen sortieren, auf die eingestellte Anzahl begrenzen
166          //und anschließend alphabetisch sortieren
167          Dictionary<string, double> topTagsDict = tagDictBlog.OrderByDescending(d =>
168           d.Value).Take(quantityTags).OrderBy(d => d.Key).ToDictionary(d => d.Key, d =>
169           d.Value);
170
171          int quantityFontSize = fontSize.Length;
172          double[] newThresholds = new double[quantityFontSize];
173          double newDelta = (Math.Log(topTagsDict.Values.Max()) -
174           Math.Log(topTagsDict.Values.Min())) / quantityFontSize;
175
176          for (int x = 0; x < quantityFontSize; x++)
177          {
178           newThresholds[x] = (x + 1) * newDelta + Math.Log(topTagsDict.Values.Min());
179          }
180
181          //foreach (string word in words)
182          foreach (KeyValuePair<string, double> tag in topTagsDict)
183          {
184           DataRow dr = tagCloudDataTableBlog.NewRow();
185           dr["HashTagName"] = tag.Key;
186           dr["FontSize"] = FontUnit.Small;
187           bool fontSet = false;
188           for (int f = 0; f < quantityFontSize; f++)
189           {
190            if ((Math.Log(tag.Value) <= newThresholds[f]) && !fontSet)
191            {
192             // assign tag to font-size/bin ‘f’
193             dr["FontSize"] = fontSize[f];
```

```
194             fontSet = true;
195           }
196         }
197         tagCloudDataTableBlog.Rows.Add(dr);
198       }
199       repeaterTagCloudBlog.DataSource = tagCloudDataTableBlog;
200       repeaterTagCloudBlog.DataBind();
201     }
202     if (tagDictAll.Count > 0)
203     {
204       DataTable tagCloudDataTableAll = new DataTable();
205       tagCloudDataTableAll.Columns.Add("HashTagName");
206       tagCloudDataTableAll.Columns.Add("FontSize", typeof(FontUnit));
207
208       FontUnit[] fontSize = { FontUnit.Small, FontUnit.Medium, FontUnit.Large, FontU-
209         nit.XLarge };
210
211       //Tags, nach Aufkommen sortieren, auf die eingestellte Anzahl begrenzen
212       //und anschließend alphabetisch sortieren
213       Dictionary<string, double> topTagsDict = tagDictAll.OrderByDescending(d =>
214         d.Value).Take(quantityTags).OrderBy(d => d.Key).ToDictionary(d => d.Key, d =>
215         d.Value);
216
217       int quantityFontSize = fontSize.Length;
218       double[] newThresholds = new double[quantityFontSize];
219       double newDelta = (Math.Log(topTagsDict.Values.Max()) -
220         Math.Log(topTagsDict.Values.Min())) / quantityFontSize;
221
222       for (int x = 0; x < quantityFontSize; x++)
223       {
224         newThresholds[x] = (x + 1) * newDelta + Math.Log(topTagsDict.Values.Min());
225       }
226
227       //foreach (string word in words)
228       foreach (KeyValuePair<string, double> tag in topTagsDict)
229       {
230         DataRow dr = tagCloudDataTableAll.NewRow();
231         dr["HashTagName"] = tag.Key;
232         dr["FontSize"] = FontUnit.Small;
233         bool fontSet = false;
234         for (int f = 0; f < quantityFontSize; f++)
235         {
236           if ((Math.Log(tag.Value) <= newThresholds[f]) && !fontSet)
237           {
238             // assign tag to font-size/bin ‘f’
239             dr["FontSize"] = fontSize[f];
240             fontSet = true;
241           }
242         }
243         tagCloudDataTableAll.Rows.Add(dr);
244         }
245       repeaterTagCloudAll.DataSource = tagCloudDataTableAll;
246       repeaterTagCloudAll.DataBind();
247     }
248     if (tagDictUser.Count > 0)
249     {
250       DataTable tagCloudDataTableUser = new DataTable();
251       tagCloudDataTableUser.Columns.Add("HashTagName");
252       tagCloudDataTableUser.Columns.Add("FontSize", typeof(FontUnit));
253
254       FontUnit[] fontSize = { FontUnit.Small, FontUnit.Medium, FontUnit.Large, FontU-
255         nit.XLarge };
256
257       //Tags, nach Aufkommen sortieren, auf die eingestellte Anzahl begrenzen
258       //und anschließend alphabetisch sortieren
```

```
259         Dictionary<string, double> topTagsDict = tagDictUser.OrderByDescending(d =>
260          d.Value).Take(quantityTags).OrderBy(d => d.Key).ToDictionary(d => d.Key, d =>
261          d.Value);
262
263         int quantityFontSize = fontSize.Length;
264         double[] newThresholds = new double[quantityFontSize];
265         double newDelta = (Math.Log(topTagsDict.Values.Max()) -
266          Math.Log(topTagsDict.Values.Min())) / quantityFontSize;
267
268         for (int x = 0; x < quantityFontSize; x++)
269         {
270          newThresholds[x] = (x + 1) * newDelta + Math.Log(topTagsDict.Values.Min());
271         }
272
273         //foreach (string word in words)
274         foreach (KeyValuePair<string, double> tag in topTagsDict)
275         {
276          DataRow dr = tagCloudDataTableUser.NewRow();
277          dr["HashTagName"] = tag.Key;
278          dr["FontSize"] = FontUnit.Small;
279          bool fontSet = false;
280          for (int f = 0; f < quantityFontSize; f++)
281          {
282           if ((Math.Log(tag.Value) <= newThresholds[f]) && !fontSet)
283           {
284            // assign tag to font-size/bin ‘f’
285            dr["FontSize"] = fontSize[f];
286            fontSet = true;
287           }
288          }
289          tagCloudDataTableUser.Rows.Add(dr);
290         }
291         repeaterTagCloudUser.DataSource = tagCloudDataTableUser;
292         repeaterTagCloudUser.DataBind();
293        }
294        }
295
296        /// <summary>
297        /// Erzeugt eine Tabelle mit den Tweets aus dem Microblog
298        /// </summary>
299        private void getMicroblogTweets()
300        {
301         //keine Suchergebnis Labe ausblenden
302         noEntriesLabel.Visible = false;
303         repeaterTweetTable.Visible = true;
304
305         //TagClouds anzeigen
306         repeaterTagCloudBlog.Visible = true;
307         repeaterTagCloudAll.Visible = false;
308         repeaterTagCloudUser.Visible = false;
309
310         //Navigationsleiste anpassen
311         navigationMicroblog.Font.Bold = true;
312         navigationAll.Font.Bold = false;
313         navigationUser.Font.Bold = false;
314
315         SPSite currentSite = SPContext.Current.Site;
316         SPServiceContext context = SPServiceContext.GetContext(currentSite);
317         UserProfileManager myUPManager = new UserProfileManager(context);
318         SPWeb currentWeb = SPContext.Current.Web;
319
320         //Überprüfen, ob Einträge im Microblog
321         if ((currentWeb.Lists["Microblog List"] != null) && (currentWeb.Lists["Microblog
322          List"].Items.Count > 0))
323         {
324          SPList tweetList = currentWeb.Lists["Microblog List"];
```

```
325         Boolean moreBtnVisible;
326         SPQuery query = new SPQuery();
327         String camlQuery = "<OrderBy><FieldRef Name='Created' Ascending='False'
328           /></OrderBy>";
329         query.Query = camlQuery;
330         DataTable tweetTable = tweetList.GetItems(query).GetDataTable();
331         tweetTable.Columns.Add("TweetID");
332         tweetTable.Columns.Add("AuthorPName");
333         tweetTable.Columns.Add("ProfileImageUrl");
334         tweetTable.Columns.Add("ProfileUrl");
335         tweetTable.Columns.Add("WebUrl");
336         tweetTable.Columns.Add("DeleteButtonVisible", typeof(Boolean));
337         tweetTable.Columns.Add("WebVisible", typeof(Boolean));
338         tweetTable.Columns.Add("Web");
339
340         String[] navigationTBHidden = NavigationTBHidden.Value.Split(new Char[] { ' '
341           });
342
343         int maxTweets = tweetEntries * Convert.ToInt32(navigationTBHidden[1]);
344         if (maxTweets >= tweetTable.Rows.Count)
345         {
346           maxTweets = tweetTable.Rows.Count;
347           moreBtnVisible = false;
348         }
349         else
350         {
351           moreBtnVisible = true;
352         }
353         for (int i = 0; i < maxTweets; i++)
354         {
355           String authorPName = GetGlobalResourceObject("microblog", "webpartus-
356             er_unknown").ToString();
357           String profileImageUrl = "~/_layouts/images/o14_person_placeholder_32.png";
358           String profileUrl = "";
359
360           //Überprüfen, ob der Nutzer  existiert
361           if (myUPManager.UserExists(tweetTable.Rows[i]["Author"].ToString()))
362           {
363             UserProfile uProfileTweet = myUPManag-
364               er.GetUserProfile(tweetTable.Rows[i]["Author"].ToString());
365             authorPName = uProfileTweet["PreferredName"].ToString();
366             profileUrl = uProfileTweet.PublicUrl.ToString();
367
368             if (uProfileTweet["PictureURL"] != null)
369             {
370               try
371               {
372                 String pictureUrl = uProfileTweet["PictureURL"].ToString();
373                 if (pictureUrl != "")
374                 {
375                   profileImageUrl = pictureUrl.Replace("MThumb", "SThumb");
376                 }
377               }
378               catch { }
379             }
380           }
381           tweetTable.Rows[i]["TweetID"] = i;
382           tweetTable.Rows[i]["AuthorPName"] = authorPName;
383           tweetTable.Rows[i]["ProfileImageUrl"] = profileImageUrl;
384           tweetTable.Rows[i]["ProfileUrl"] = profileUrl;
385           tweetTable.Rows[i]["WebUrl"] = "";
386           tweetTable.Rows[i]["DeleteButtonVisible"] = false;
387           tweetTable.Rows[i]["WebVisible"] = false;
388           tweetTable.Rows[i]["Web"] = "";
389         }
390         moreTweets.Visible = moreBtnVisible;
```

```
391
392         repeaterTweetTable.DataSource = getLatestRowsDataSet(maxTweets, tweetTable,
393           null);
394         repeaterTweetTable.DataBind();
395       }
396       else
397       {
398         repeaterTweetTable.Visible = false;
399         moreTweets.Visible = false;
400       }
401     }
402
403     /// <summary>
404     /// Erzeugt eine Tabelle mit den Tweets aus allen verfügbaren Microblogs
405     /// </summary>
406     private void getAllTweets()
407     {
408       //keine Suchergebnis Labe ausblenden
409       noEntriesLabel.Visible = false;
410       repeaterTweetTable.Visible = true;
411
412       //TagClouds anzeigen
413       repeaterTagCloudBlog.Visible = false;
414       repeaterTagCloudAll.Visible = true;
415       repeaterTagCloudUser.Visible = false;
416
417       //Navigationsleiste anpassen
418       navigationMicroblog.Font.Bold = false;
419       navigationAll.Font.Bold = true;
420       navigationUser.Font.Bold = false;
421
422       SPSite currentSite = SPContext.Current.Site;
423       String siteUrl = currentSite.Url;
424       SPServiceContext context = SPServiceContext.GetContext(currentSite);
425       UserProfileManager myUPManager = new UserProfileManager(context);
426       String loginName = SPContext.Current.Web.CurrentUser.LoginName;
427       DataTable mergeTable = new DataTable();
428
429       //Auflistung aller Microbloglisten (ElevatedPrivileges notwendig wegen AllWebs)
430       SPSecurity.RunWithElevatedPrivileges(delegate()
431       {
432         SPSite cSite = new SPSite(siteUrl);
433         //Die Listen für Alle Seiten in tweetTable mergen
434         foreach (SPWeb web in cSite.AllWebs)
435         {
436           //Überprüfen, ob der Nutzer auf die Liste zugreifen darf
437           if ((checkListExistence(web, "Microblog List") &&
438             (web.DoesUserHavePermissions(loginName, SPBasePermissions.ViewPages))))
439           {
440             DataTable tempTable = new DataTable();
441             tempTable = web.Lists["Microblog List"].Items.GetDataTable();
442             if ((tempTable != null) && (tempTable.Rows.Count > 0))
443             {
444               tempTable.Columns.Add("Web");
445               tempTable.Columns.Add("WebUrl");
446               for (int i = 0; i < tempTable.Rows.Count; i++)
447               {
448                 tempTable.Rows[i]["Web"] = web.Title;
449                 tempTable.Rows[i]["WebUrl"] = web.Url;
450               }
451               mergeTable.Merge(tempTable);
452             }
453           }
454         }
455       }
456       );
```

```
457         if ((mergeTable != null) && (mergeTable.Rows.Count > 0))
458         {
459           mergeTable.DefaultView.Sort = "Created DESC";
460           DataTable tweetTable = mergeTable.DefaultView.ToTable();
461           mergeTable.Dispose();
462           tweetTable.Columns.Add("TweetID");
463           tweetTable.Columns.Add("AuthorPName");
464           tweetTable.Columns.Add("ProfileImageUrl");
465           tweetTable.Columns.Add("ProfileUrl");
466           tweetTable.Columns.Add("DeleteButtonVisible", typeof(Boolean));
467           tweetTable.Columns.Add("WebVisible", typeof(Boolean));
468
469           Boolean moreBtnVisible;
470           String[] navigationTBHidden = NavigationTBHidden.Value.Split(new Char[] { ' '
471             });
472
473           int maxTweets = tweetEntries * Convert.ToInt32(navigationTBHidden[1]);
474           if (maxTweets >= tweetTable.Rows.Count)
475           {
476             maxTweets = tweetTable.Rows.Count;
477             moreBtnVisible = false;
478           }
479           else
480           {
481             moreBtnVisible = true;
482           }
483           for (int i = 0; i < maxTweets; i++)
484           {
485             String authorPName = GetGlobalResourceObject("microblog", "webpartus-
486               er_unknown").ToString();
487             String profileImageUrl = "~/_layouts/images/o14_person_placeholder_32.png";
488             String profileUrl = "";
489
490             //Überprüfen, ob der Nutzer  existiert
491             if (myUPManager.UserExists(tweetTable.Rows[i]["Author"].ToString()))
492             {
493               UserProfile uProfileTweet = myUPManag-
494                 er.GetUserProfile(tweetTable.Rows[i]["Author"].ToString());
495               authorPName = uProfileTweet["PreferredName"].ToString();
496               profileUrl = uProfileTweet.PublicUrl.ToString();
497
498               if (uProfileTweet["PictureURL"] != null)
499               {
500                 try
501                 {
502                   String pictureUrl = uProfileTweet["PictureURL"].ToString();
503                   if (pictureUrl != "")
504                   {
505                     profileImageUrl = pictureUrl.Replace("MThumb", "SThumb");
506                   }
507                 }
508                 catch { }
509               }
510             }
511             tweetTable.Rows[i]["TweetID"] = i;
512             tweetTable.Rows[i]["AuthorPName"] = authorPName;
513             tweetTable.Rows[i]["ProfileImageUrl"] = profileImageUrl;
514             tweetTable.Rows[i]["ProfileUrl"] = profileUrl;
515             tweetTable.Rows[i]["DeleteButtonVisible"] = false;
516             tweetTable.Rows[i]["WebVisible"] = true;
517           }
518           moreTweets.Visible = moreBtnVisible;
519
520           repeaterTweetTable.DataSource = getLatestRowsDataSet(maxTweets, tweetTable,
521             null);
522           repeaterTweetTable.DataBind();
```

```
523         }
524         else
525         {
526           repeaterTweetTable.Visible = false;
527           moreTweets.Visible = false;
528         }
529       }
530
531       /// <summary>
532       /// Erzeugt eine Tabelle mit allen Tweets des Users und allen an den User gerichte-
533         ten
534       /// </summary>
535       private void getUserTweets()
536       {
537         //keine Suchergebnis Labe ausblenden
538         noEntriesLabel.Visible = false;
539         repeaterTweetTable.Visible = true;
540
541         //TagClouds anzeigen
542         repeaterTagCloudBlog.Visible = false;
543         repeaterTagCloudAll.Visible = false;
544         repeaterTagCloudUser.Visible = true;
545
546         //Navigationsleiste anpassen
547         navigationMicroblog.Font.Bold = false;
548         navigationAll.Font.Bold = false;
549         navigationUser.Font.Bold = true;
550
551         SPSite currentSite = SPContext.Current.Site;
552         String siteUrl = currentSite.Url;
553         SPServiceContext context = SPServiceContext.GetContext(currentSite);
554         UserProfileManager myUPManager = new UserProfileManager(context);
555         String loginName = SPContext.Current.Web.CurrentUser.LoginName;
556
557         UserProfile uProfileCurrent = ProfileLoader.GetProfileLoader().GetUserProfile();
558         String userTag = "@" + uProfileCurrent["PreferredName"].ToString();
559
560         SPQuery query = new SPQuery();
561         query.ExpandUserField = false;
562         query.Query = "<Where><Or>" + "<Eq><FieldRef Name='Author' /><Value Type='User'>"
563           + SPContext.Current.Web.SiteUsers[uProfileCurrent["AccountName"].ToString()] +
564           "</Value></Eq>" + "<Contains><FieldRef Name='Tweet' /><Value Type='Text'>" +
565           userTag + "</Value></Contains>" + "</Or></Where>";
566         DataTable mergeTable = new DataTable();
567
568         //Auflistung aller Microbloglisten (ElevatedPrivileges notwendig wegen AllWebs)
569         SPSecurity.RunWithElevatedPrivileges(delegate()
570         {
571           //Die Listen für Alle Seiten in tweetTable mergen
572           SPSite cSite = new SPSite(siteUrl);
573           foreach (SPWeb web in cSite.AllWebs)
574           {
575             //Auswahl aller Seiten, die vom Typ Microblog List sind und auf die der Nutzer
576               zugreifen darf
577             if ((checkListExistence(web, "Microblog List") &&
578               (web.DoesUserHavePermissions(loginName, SPBasePermissions.ViewPages))))
579             {
580               DataTable tempTable = new DataTable();
581               tempTable = web.Lists["Microblog List"].GetItems(query).GetDataTable();
582               if ((tempTable != null) && (tempTable.Rows.Count > 0))
583               {
584                 tempTable.Columns.Add("Web");
585                 tempTable.Columns.Add("WebUrl");
586                 for (int i = 0; i < tempTable.Rows.Count; i++)
587                 {
588                   tempTable.Rows[i]["Web"] = web.Title;
```

```
589                 tempTable.Rows[i]["WebUrl"] = web.Url;
590               }
591               mergeTable.Merge(tempTable);
592             }
593           }
594         }
595       }
596       );
597       if ((mergeTable != null) && (mergeTable.Rows.Count > 0))
598       {
599         mergeTable.DefaultView.Sort = "Created DESC";
600         DataTable tweetTable = mergeTable.DefaultView.ToTable();
601         mergeTable.Dispose();
602         tweetTable.Columns.Add("TweetID");
603         tweetTable.Columns.Add("AuthorPName");
604         tweetTable.Columns.Add("ProfileImageUrl");
605         tweetTable.Columns.Add("ProfileUrl");
606         tweetTable.Columns.Add("DeleteButtonVisible", typeof(Boolean));
607         tweetTable.Columns.Add("WebVisible", typeof(Boolean));
608
609         Boolean moreBtnVisible;
610         String[] navigationTBHidden = NavigationTBHidden.Value.Split(new Char[] { ' '
611           });
612
613         int maxTweets = tweetEntries * Convert.ToInt32(navigationTBHidden[1]);
614         if (maxTweets >= tweetTable.Rows.Count)
615         {
616           maxTweets = tweetTable.Rows.Count;
617           moreBtnVisible = false;
618         }
619         else
620         {
621           moreBtnVisible = true;
622         }
623         for (int i = 0; i < maxTweets; i++)
624         {
625           String authorPName = GetGlobalResourceObject("microblog", "webpartus-
626             er_unknown").ToString();
627           String profileImageUrl = "~/_layouts/images/o14_person_placeholder_32.png";
628           String profileUrl = "";
629           bool deleteButtonVisible = false;
630
631           //Überprüfen, ob der Nutzer  existiert
632           if (myUPManager.UserExists(tweetTable.Rows[i]["Author"].ToString()))
633           {
634             UserProfile uProfileTweet = myUPManag-
635               er.GetUserProfile(tweetTable.Rows[i]["Author"].ToString());
636             if (tweetTable.Rows[i]["Author"].ToString() == uProfileCur-
637               rent["AccountName"].ToString())
638             {
639               deleteButtonVisible = true;
640             }
641             authorPName = uProfileTweet["PreferredName"].ToString();
642             profileUrl = uProfileTweet.PublicUrl.ToString();
643
644             if (uProfileTweet["PictureURL"] != null)
645             {
646               try
647               {
648                 String pictureUrl = uProfileTweet["PictureURL"].ToString();
649                 if (pictureUrl != "")
650                 {
651                   profileImageUrl = pictureUrl.Replace("MThumb", "SThumb");
652                 }
653               }
654               catch { }
```

```
655             }
656           }
657           tweetTable.Rows[i]["TweetID"] = i;
658           tweetTable.Rows[i]["AuthorPName"] = authorPName;
659           tweetTable.Rows[i]["ProfileImageUrl"] = profileImageUrl;
660           tweetTable.Rows[i]["ProfileUrl"] = profileUrl;
661           tweetTable.Rows[i]["DeleteButtonVisible"] = deleteButtonVisible;
662           tweetTable.Rows[i]["WebVisible"] = true;
663         }
664         moreTweets.Visible = moreBtnVisible;
665
666         repeaterTweetTable.DataSource = getLatestRowsDataSet(maxTweets, tweetTable,
667           null);
668         repeaterTweetTable.DataBind();
669       }
670       else
671       {
672         repeaterTweetTable.Visible = false;
673         moreTweets.Visible = false;
674       }
675     }
676
677     /// <summary>
678     /// Durchsucht alle Tweets aus den verfügbaren Microblogs und bildet das Ergebnis
679       in der tweetTabelle ab
680     /// </summary>
681     private void getSearchTweets(String searchString, String navigationScope)
682     {
683       //keine Suchergebnis Labe ausblenden
684       noEntriesLabel.Visible = false;
685       repeaterTweetTable.Visible = true;
686
687       SPSite currentSite = SPContext.Current.Site;
688       String siteUrl = currentSite.Url;
689       SPServiceContext context = SPServiceContext.GetContext(currentSite);
690       UserProfileManager myUPManager = new UserProfileManager(context);
691       String loginName = SPContext.Current.Web.CurrentUser.LoginName;
692       DataTable mergeTable = new DataTable();
693
694       SPQuery query = new SPQuery();
695       query.ExpandUserField = false;
696
697       if (navigationScope != "navigationMicroblog")
698       {
699         switch (navigationScope)
700         {
701           case "navigationAll":
702             query.Query = "<Where><Contains><FieldRef Name='Tweet' /><Value Type='Text'>"
703               + searchString + "</Value></Contains></Where>";
704             break;
705           case "navigationUser":
706             UserProfile uProfileCurrent = ProfileLoad-
707               er.GetProfileLoader().GetUserProfile();
708             String userTag = "@" + uProfileCurrent["PreferredName"].ToString();
709             query.Query = "<Where><And><Or>"
710             + "<Eq><FieldRef Name='Author' /><Value Type='User'>" + SPCon-
711               text.Current.Web.SiteUsers[uProfileCurrent["AccountName"].ToString()] +
712               "</Value></Eq>" + "<Contains><FieldRef Name='Tweet' /><Value Type='Text'>"
713               + userTag + "</Value></Contains></Or>" + "<Contains><FieldRef Name='Tweet'
714               /><Value Type='Text'>" + searchString +"</Value></Contains></And></Where>";
715             break;
716           default:
717             query.Query = "<Where><Contains><FieldRef Name='Tweet' /><Value Type='Text'>"
718               + searchString + "</Value></Contains></Where>";
719             break;
720         }
```

```
721
722         //Auflistung aller Microbloglisten (ElevatedPrivileges notwendig wegen AllWebs)
723         SPSecurity.RunWithElevatedPrivileges(delegate()
724         {
725           SPSite cSite = new SPSite(siteUrl);
726           //Die Listen für Alle Seiten in tweetTable mergen
727           foreach (SPWeb web in cSite.AllWebs)
728           {
729             //Überprüfen, ob der Nutzer auf die Liste zugreifen darf
730             if ((checkListExistence(web, "Microblog List") &&
731               (web.DoesUserHavePermissions(loginName, SPBasePermissions.ViewPages))))
732             {
733               DataTable tempTable = new DataTable();
734               tempTable = web.Lists["Microblog List"].GetItems(query).GetDataTable(); ;
735               if ((tempTable != null) && (tempTable.Rows.Count > 0))
736               {
737                 tempTable.Columns.Add("Web");
738                 tempTable.Columns.Add("WebUrl");
739                 for (int i = 0; i < tempTable.Rows.Count; i++)
740                 {
741                   tempTable.Rows[i]["Web"] = web.Title;
742                   tempTable.Rows[i]["WebUrl"] = web.Url;
743                 }
744                 mergeTable.Merge(tempTable);
745               }
746             }
747           }
748         }
749         );
750       }
751       else
752       {
753         SPWeb currentWeb = SPContext.Current.Web;
754
755         //Überprüfen, ob Einträge im Microblog
756         if ((currentWeb.Lists["Microblog List"] != null) && (currentWeb.Lists["Microblog
757           List"].Items.Count > 0))
758         {
759           SPList tweetList = currentWeb.Lists["Microblog List"];
760           String camlQuery = "<Where><Contains><FieldRef Name='Tweet' /><Value
761             Type='Text'>" + searchString + "</Value></Contains></Where>" + "<Order-
762             By><FieldRef Name='Created' Ascending='False' /></OrderBy>";
763           query.Query = camlQuery;
764           mergeTable = tweetList.GetItems(query).GetDataTable();
765           if (mergeTable != null)
766           {
767             mergeTable.Columns.Add("Web");
768             mergeTable.Columns.Add("WebUrl");
769           }
770         }
771       }
772       if ((mergeTable != null) && (mergeTable.Rows.Count > 0))
773       {
774         mergeTable.DefaultView.Sort = "Created DESC";
775         DataTable tweetTable = mergeTable.DefaultView.ToTable();
776         mergeTable.Dispose();
777         tweetTable.Columns.Add("TweetID");
778         tweetTable.Columns.Add("AuthorPName");
779         tweetTable.Columns.Add("ProfileImageUrl");
780         tweetTable.Columns.Add("ProfileUrl");
781         tweetTable.Columns.Add("DeleteButtonVisible", typeof(Boolean));
782         tweetTable.Columns.Add("WebVisible", typeof(Boolean));
783
784         Boolean moreBtnVisible;
785         String[] navigationTBHidden = NavigationTBHidden.Value.Split(new Char[] { ' '
786           });
```

```
787
788         int maxTweets = tweetEntries * Convert.ToInt32(navigationTBHidden[1]);
789         if (maxTweets >= tweetTable.Rows.Count)
790         {
791           maxTweets = tweetTable.Rows.Count;
792           moreBtnVisible = false;
793         }
794         else
795         {
796           moreBtnVisible = true;
797         }
798         for (int i = 0; i < maxTweets; i++)
799         {
800           String authorPName = GetGlobalResourceObject("microblog", "webpartus-
801             er_unknown").ToString();
802           String profileImageUrl = "~/_layouts/images/o14_person_placeholder_32.png";
803           String profileUrl = "";
804
805           //Überprüfen, ob der Nutzer  existiert
806           if (myUPManager.UserExists(tweetTable.Rows[i]["Author"].ToString()))
807           {
808             UserProfile uProfileTweet = myUPManag-
809               er.GetUserProfile(tweetTable.Rows[i]["Author"].ToString());
810             authorPName = uProfileTweet["PreferredName"].ToString();
811             profileUrl = uProfileTweet.PublicUrl.ToString();
812
813             if (uProfileTweet["PictureURL"] != null)
814             {
815               try
816               {
817                 String pictureUrl = uProfileTweet["PictureURL"].ToString();
818                 if (pictureUrl != "")
819                 {
820                   profileImageUrl = pictureUrl.Replace("MThumb", "SThumb");
821                 }
822               }
823               catch { }
824             }
825           }
826           tweetTable.Rows[i]["TweetID"] = i;
827           tweetTable.Rows[i]["AuthorPName"] = authorPName;
828           tweetTable.Rows[i]["ProfileImageUrl"] = profileImageUrl;
829           tweetTable.Rows[i]["ProfileUrl"] = profileUrl;
830           tweetTable.Rows[i]["DeleteButtonVisible"] = false;
831           tweetTable.Rows[i]["WebVisible"] = true;
832         }
833         moreTweets.Visible = moreBtnVisible;
834
835         repeaterTweetTable.DataSource = getLatestRowsDataSet(maxTweets, tweetTable,
836           searchString);
837         repeaterTweetTable.DataBind();
838       }
839       else
840       {
841         noEntriesLabel.Visible = true;
842         repeaterTweetTable.Visible = false;
843         moreTweets.Visible = false;
844       }
845     }
846
847     /// <summary>
848     /// notwendig für den JavaScript für Zeichenlimit-Anzeige
849     /// </summary>
850     /// <param name="tweetLengthLimit">eingestelltes Zeichenlimit</param>
851     private void setInputLimit(int tweetLengthLimit)
852     {
```

```
853         tweetInput.Attributes.Add("MaxLength", tweetLengthLimit.ToString());
854         tweetInput.MaxLength = tweetLengthLimit;
855         tweetInputLeft.Text = tweetInput.MaxLength.ToString();
856       }
857
858       /// <summary>
859       /// Erzeugt einen entsprechen Tweet-Eintrag in der Microblog List Liste
860       /// </summary>
861       /// <param name="tweetMessage">Text, der als Tweet eingetragen werden soll</param>
862       private void setTweet(String tweetMessage)
863       {
864         SPSite currentSite = SPContext.Current.Site;
865         SPServiceContext context = SPServiceContext.GetContext(currentSite);
866         UserProfileManager myUPManager = new UserProfileManager(context);
867         UserProfile uProfileCurrent = ProfileLoader.GetProfileLoader().GetUserProfile();
868
869         String loginName = uProfileCurrent["AccountName"].ToString();
870         String webUrl = SPContext.Current.Web.Url;
871         SPSecurity.RunWithElevatedPrivileges(delegate()
872         {
873           using (SPSite siteCollection = new SPSite(webUrl))
874           {
875             using (SPWeb web = siteCollection.OpenWeb())
876             {
877               try
878               {
879                 web.AllowUnsafeUpdates = true;
880                 SPList tweetList = web.Lists["Microblog List"];
881                 SPListItem newItem = tweetList.Items.Add();
882                 newItem["Tweet"] = tweetMessage;
883                 newItem["Author"] = web.SiteUsers[loginName];
884                 newItem["HashTags"] = getHashTags(tweetMessage);
885                 newItem.Update();
886                 web.AllowUnsafeUpdates = false;
887               }
888               catch (Exception e)
889               {
890                 Console.WriteLine(e.ToString());
891               }
892             }
893           }
894         });
895       }
896
897       private String getHashTags(String tweetMessage)
898       {
899         String hashTagString = String.Empty;
900
901         String[] stringsWithoutSpaces = tweetMessage.Split(new Char[] { ' ' });
902
903         Regex regularExpressionHashTag = new Regex("^#\\w+");
904         List<String> hashTagMatches = new List<String>();
905
906         for (int i = 0; i < stringsWithoutSpaces.Length; i++)
907         {
908           if ((regularExpressionHashTag.IsMatch(stringsWithoutSpaces[i])) && (hashTag-
909             Matches.Contains(stringsWithoutSpaces[i].ToString()) != true))
910           {
911             hashTagMatches.Add(stringsWithoutSpaces[i].ToString().Substring(1));
912           }
913         }
914
915         hashTagString = String.Join(" ", hashTagMatches.ToArray());
916
917         return hashTagString;
918       }
```

```
919
920         /// <summary>Überprüft, ob Sharepoint-Liste mit einem bestimmten Namen exis-
921           tiert</summary>
922         /// <param name="web">SPWeb in dem die Liste existieren soll</param>
923         /// <param name="listName">Name der Sharepoint-Liste</param>
924         static bool checkListExistence(SPWeb web, String listName)
925         {
926           return web.Lists.Cast<SPList>().Any(list => String.Equals(list.Title, listName));
927         }
928
929         /// <summary>Erzeugt eine DataSet mit rowCount Rows & Relation mit der
930           MessageTable</summary>
931         /// <param name="rowCount">Anzahl der Zeilen, die die DataTable beinhalten
932           soll</param>
933         /// <param name="dt">Die DataTable die verkürzt werden soll</param>
934         private DataSet getLatestRowsDataSet(int rowCount, DataTable dt, String search-
935           String)
936         {
937           try
938           {
939             DataTable newTweetTable = dt.Clone();
940             DataTable messageTable = new DataTable();
941             if (rowCount > dt.Rows.Count)
942             {
943               rowCount = dt.Rows.Count;
944             }
945             for (int i = 0; i < rowCount; i++)
946             {
947               DataRow tr = dt.Rows[i];
948               newTweetTable.ImportRow(tr);
949               messageTa-
950                 ble.Merge(generateTweetMessage(SPEncode.HtmlEncode(dt.Rows[i]["Tweet"].ToStri
951                 ng()), searchString, dt.Rows[i]["TweetID"].ToString()));
952             }
953             DataSet ds = new DataSet();
954             ds.Tables.Add(newTweetTable);
955             ds.Tables.Add(messageTable);
956             ds.Relations.Add(new DataRelation("NestedMessages",
957               ds.Tables[0].Columns["TweetID"], ds.Tables[1].Columns["TweetID"]));
958             return ds;
959           }
960           catch
961           {
962             return new DataSet();
963           }
964         }
965
966         /// <summary>Erzeugt einen String mit HTML-Controls, die Tags und einen evtl.
967           Suchterm herrausfiltern</summary>
968         /// <param name="tweetString">Eintrag der durchsucht werden soll</param>
969         /// <param name="searchString">Suchterm, der hervorgehoben werden soll</param>
970         private static DataTable generateTweetMessage(String tweetString, String search-
971           String, String tweetID)
972         {
973           DataTable messageTable = new DataTable();
974           messageTable.Columns.Add("TweetID");
975           messageTable.Columns.Add("UserTag");
976           messageTable.Columns.Add("UserTagUrl");
977           messageTable.Columns.Add("UserTagVisible", typeof(Boolean));
978           messageTable.Columns.Add("HashTag");
979           messageTable.Columns.Add("HashTagVisible", typeof(Boolean));
980           messageTable.Columns.Add("TweetText");
981           messageTable.Columns.Add("TweetTextVisible", typeof(Boolean));
982
983           String boldTweet = tweetString;
984
```

```
985         if ((searchString != "") && (searchString != null))
986         {
987           String[] dummyStringArray = { searchString };
988           String[] subStrings = Regex.Split(tweetString, searchString, RegexOp-
989             tions.IgnoreCase);
990 
991           boldTweet = String.Join("<b>" + searchString + "</b>", subStrings);
992         }
993 
994         String[] stringsWithoutSpaces = boldTweet.Split(new Char[] { ' ' });
995 
996         Regex regularExpressionUserTag = new Regex("^@\\w+");
997         Regex regularExpressionHashTag = new Regex("^#\\w+");
998 
999         SPSite currentSite = SPContext.Current.Site;
1000        SPServiceContext context = SPServiceContext.GetContext(currentSite);
1001        UserProfileManager myUPManager = new UserProfileManager(context);
1002 
1003        for (int i = 0; i < stringsWithoutSpaces.Length; i++)
1004        {
1005          //Usertags herausfiltern und Hashtags herausfiltern
1006          if ((regularExpressionUserTag.IsMatch(stringsWithoutSpaces[i])) || (regularEx-
1007            pressionHashTag.IsMatch(stringsWithoutSpaces[i])))
1008          {
1009            DataRow dr = messageTable.NewRow();
1010            dr["TweetID"] = tweetID;
1011            dr["TweetText"] = "";
1012            dr["TweetTextVisible"] = false;
1013 
1014            //Hashtag in DataTable
1015            if (regularExpressionHashTag.IsMatch(stringsWithoutSpaces[i]))
1016            {
1017              dr["UserTag"] = "";
1018              dr["UserTagUrl"] = "";
1019              dr["UserTagVisible"] = false;
1020              dr["HashTag"] = stringsWithoutSpaces[i];
1021              dr["HashTagVisible"] = true;
1022            }
1023            else
1024            {
1025              //Usertag in DataTable
1026              String pName = stringsWithoutSpaces[i].Substring(1);
1027              if (myUPManager.UserExists(pName))
1028              {
1029                UserProfile uProfileTweet = myUPManager.GetUserProfile(pName);
1030                dr["UserTag"] = stringsWithoutSpaces[i];
1031                dr["UserTagUrl"] = uProfileTweet.PublicUrl.ToString();
1032                dr["UserTagVisible"] = true;
1033                dr["HashTag"] = "";
1034                dr["HashTagVisible"] = false;
1035              }
1036              else
1037              {
1038                dr["UserTag"] = "";
1039                dr["UserTagUrl"] = "";
1040                dr["UserTagVisible"] = false;
1041                dr["HashTag"] = "";
1042                dr["HashTagVisible"] = false;
1043                dr["TweetText"] = stringsWithoutSpaces[i] + " ";
1044                dr["TweetTextVisible"] = true;
1045              }
1046            }
1047            messageTable.Rows.Add(dr);
1048          }
1049          else
1050          {
```

```
1051          if ((messageTable.Rows.Count > 0) && (messageTable.Rows != null))
1052          {
1053            int rowIndex = messageTable.Rows.Count - 1;
1054            messageTable.Rows[rowIndex]["TweetTextVisible"] = true;
1055            messageTable.Rows[rowIndex]["TweetText"] = messageTa-
1056              ble.Rows[rowIndex]["TweetText"] + " " + stringsWithoutSpaces[i];
1057          }
1058          else
1059          {
1060            DataRow dr = messageTable.NewRow();
1061            dr["TweetID"] = tweetID;
1062            dr["UserTag"] = "";
1063            dr["UserTagUrl"] = "";
1064            dr["UserTagVisible"] = false;
1065            dr["HashTag"] = "";
1066            dr["HashTagVisible"] = false;
1067            dr["TweetTextVisible"] = true;
1068            dr["TweetText"] = stringsWithoutSpaces[i];
1069            messageTable.Rows.Add(dr);
1070          }
1071        }
1072      }
1073      return messageTable;
1074    }
1075
1076    // DataBinding für den NestedRepeater
1077    protected void repeaterTweetTable_ItemDataBound(object sender, Sys-
1078      tem.Web.UI.WebControls.RepeaterItemEventArgs e)
1079    {
1080      DataRowView dv = e.Item.DataItem as DataRowView;
1081      if (dv != null)
1082      {
1083        Repeater nestedRepeater = e.Item.FindControl("repeaterTweetMessage") as Repea-
1084          ter;
1085        if (nestedRepeater != null)
1086        {
1087          nestedRepeater.DataSource = dv.CreateChildView("NestedMessages");
1088          nestedRepeater.DataBind();
1089        }
1090      }
1091    }
1092
1093    // DeleteButton zum Löschen einer TweetMessage wird geklickt
1094    protected void repeaterTweetTable_ItemCommand(object source, RepeaterCommandEven-
1095      tArgs e)
1096    {
1097      if (e.CommandName == "DeletePost")
1098      {
1099        if (e.CommandArgument != null)
1100        {
1101          deletePost(e.CommandArgument.ToString());
1102          getUserTweets();
1103        }
1104      }
1105    }
1106
1107    // Tag in einer TweetMessage wird geklickt
1108    protected void repeaterTweetMessage_ItemCommand(object source, RepeaterCommandEven-
1109      tArgs e)
1110    {
1111      if (e.CommandName == "HashTag")
1112      {
1113        if (e.CommandArgument != null)
1114        {
1115          String navigationScope = "";
```

```
1116          String[] navigationTBHidden = NavigationTBHidden.Value.Split(new Char[] { ' '
1117            });
1118          if (navigationTBHidden[0] != "navigationSearch")
1119          {
1120            navigationScope = navigationTBHidden[0];
1121            NavigationTBHidden.Value = "navigationSearch 1 " + navigationTBHidden[0];
1122          }
1123          else
1124          {
1125            navigationScope = navigationTBHidden[2];
1126          }
1127          getSearchTweets(e.CommandArgument.ToString(), navigationScope);
1128        }
1129      }
1130    }
1131
1132    // Tag in der TagCloud wird geklickt
1133    protected void repeaterTagCloud_ItemCommand(object source, RepeaterCommandEventArgs
1134      e)
1135    {
1136      if (e.CommandName == "HashTag")
1137      {
1138        if (e.CommandArgument != null)
1139        {
1140          String navigationScope = "";
1141          String[] navigationTBHidden = NavigationTBHidden.Value.Split(new Char[] { ' '
1142            });
1143          if (navigationTBHidden[0] != "navigationSearch")
1144          {
1145            navigationScope = navigationTBHidden[0];
1146            NavigationTBHidden.Value = "navigationSearch 1 " + navigationTBHidden[0];
1147            }
1148          else
1149          {
1150            navigationScope = navigationTBHidden[2];
1151          }
1152          getSearchTweets("#" + e.CommandArgument.ToString(), navigationScope);
1153        }
1154      }
1155    }
1156
1157    protected void tweetSubmit_Click(object sender, EventArgs e)
1158    {
1159      if (tweetInput.Text != "" && tweetInput.Text.Length <= tweetLengthLimit)
1160      {
1161        setTweet(tweetInput.Text);
1162      }
1163      tweetSubmit.Enabled = false;
1164      tweetInputLeft.Text = tweetInput.MaxLength.ToString();
1165      tweetInput.Text = "";
1166      getMicroblogTweets();
1167    }
1168
1169    protected void moreTweets_Click(object sender, EventArgs e)
1170    {
1171      String[] navigationTBHidden = NavigationTBHidden.Value.Split(new Char[] { ' ' });
1172      String navigationScope = navigationTBHidden[0];
1173      NavigationTBHidden.Value = navigationTBHidden[0] + " " + (Con-
1174        vert.ToInt32(navigationTBHidden[1]) + 1).ToString();
1175
1176      switch (navigationScope)
1177      {
1178        case "navigationMicroblog":
1179          getMicroblogTweets();
1180          break;
1181        case "navigationAll":
```

```
1182            getAllTweets();
1183            break;
1184          case "navigationUser":
1185            getUserTweets();
1186            break;
1187          case "navigationSearch":
1188            {
1189              getSearchTweets(searchTextBox.Text, navigationTBHidden[2]);
1190            }
1191            break;
1192          default:
1193            getMicroblogTweets();
1194            break;
1195        }
1196      }
1197
1198      // Suche wurde über die searchTextBox abgeschickt
1199      protected void searchButton_Click(object sender, ImageClickEventArgs e)
1200      {
1201        String[] navigationTBHidden = NavigationTBHidden.Value.Split(new Char[] { ' ' });
1202        if (navigationTBHidden[0] != "navigationSearch")
1203        {
1204          NavigationTBHidden.Value = "navigationSearch 1 " + navigationTBHidden[0];
1205          getSearchTweets(searchTextBox.Text, navigationTBHidden[0]);
1206        }
1207        else
1208        {
1209          getSearchTweets(searchTextBox.Text, navigationTBHidden[2]);
1210        }
1211      }
1212
1213      #region ClickEvents der NavigationsLinkButtons
1214      protected void navigationMicroblog_Click(object sender, EventArgs e)
1215      {
1216        repeaterTagCloudAll.Visible = false;
1217        NavigationTBHidden.Value = "navigationMicroblog 1";
1218        getMicroblogTweets();
1219      }
1220
1221      protected void navigationAll_Click(object sender, EventArgs e)
1222      {
1223        repeaterTagCloudAll.Visible = true;
1224        NavigationTBHidden.Value = "navigationAll 1";
1225        getAllTweets();
1226      }
1227
1228      protected void navigationUser_Click(object sender, EventArgs e)
1229      {
1230        NavigationTBHidden.Value = "navigationUser 1";
1231        getUserTweets();
1232      }
1233      #endregion
1234
1235      /// <summary>Funktion zum löschen der Nutzereinträge</summary>
1236      /// <param name="commandArgument">Beinhaltet die webUrl und ID des jeweiligen Ein-
1237        trags</param>
1238      private void deletePost(String commandArgument)
1239      {
1240        String[] commandArgs = commandArgument.Split(new char[] { ',' });
1241
1242        SPSecurity.RunWithElevatedPrivileges(delegate()
1243        {
1244          using (SPSite siteCollection = new SPSite(commandArgs[0]))
1245          {
1246            using (SPWeb web = siteCollection.OpenWeb())
1247            {
```

```
1248            try
1249            {
1250              web.AllowUnsafeUpdates = true;
1251              web.Lists["Microblog
1252                List"].GetItemById(Convert.ToInt32(commandArgs[1])).Delete();
1253              web.Lists["Microblog List"].Update();
1254              web.AllowUnsafeUpdates = false;
1255            }
1256            catch (Exception e)
1257            {
1258              Console.WriteLine(e.ToString());
1259            }
1260          }
1261        }
1262      });
1263    }
1264  }
1265 }
```

Anhang E – Quellcode MicroBlogVisualWebPartUserControl.ascx

```
<%@ Assembly Name="$SharePoint.Project.AssemblyFullName$" %>
<%@ Assembly Name="Microsoft.Web.CommandUI, Version=14.0.0.0, Culture=neutral,
 PublicKeyToken=71e9bce111e9429c" %>
<%@ Register TagPrefix="SharePoint" Namespace="Microsoft.SharePoint.WebControls"
 Assembly="Microsoft.SharePoint, Version=14.0.0.0, Culture=neutral,
 PublicKeyToken=71e9bce111e9429c" %>
<%@ Register TagPrefix="Utilities" Namespace="Microsoft.SharePoint.Utilities"
 Assembly="Microsoft.SharePoint, Version=14.0.0.0, Culture=neutral,
 PublicKeyToken=71e9bce111e9429c" %>
<%@ Register TagPrefix="asp" Namespace="System.Web.UI"
 Assembly="System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
 PublicKeyToken=31bf3856ad364e35" %>
<%@ Import Namespace="Microsoft.SharePoint" %>
<%@ Register TagPrefix="WebPartPages" Namespace="Microsoft.SharePoint.WebPartPages"
 Assembly="Microsoft.SharePoint, Version=14.0.0.0, Culture=neutral,
 PublicKeyToken=71e9bce111e9429c" %>
<%@ Control Language="C#" AutoEventWireup="true"
 CodeBehind="MicroBlogVisualWebPartUserControl.ascx.cs"
 Inherits="MicroblogSolution.MicroBlogVisualWebPart.MicroBlogVisualWebPartUserControl
 " %>
<script type="text/javascript">
//<![CDATA[
 function Count(text) {
  var maxlength = new Number(text.attributes["MaxLength"].value);
  if (document.getElementById('<%=tweetInput.ClientID%>').value.length > maxlength) {
   document.getElementById('<%=tweetSubmit.ClientID%>').disabled = true;
   document.getElementById('<%=tweetInputLeft.ClientID%>').innerHTML = (maxlength -
    document.getElementById('<%=tweetInput.ClientID%>').value.length);
   document.getElementById('<%=tweetInputLeft.ClientID%>').style.color = "red";
  }
  else {
   if (document.getElementById('<%=tweetInput.ClientID%>').value.length == 0) {
    document.getElementById('<%=tweetSubmit.ClientID%>').disabled = true;
   }
   else {
    document.getElementById('<%=tweetSubmit.ClientID%>').disabled = false;
   }
   document.getElementById('<%=tweetInputLeft.ClientID%>').innerHTML = maxlength -
    document.getElementById('<%=tweetInput.ClientID%>').value.length;
   document.getElementById('<%=tweetInputLeft.ClientID%>').style.color = "";
  }
 }

 function SearchText(sb) {
  if (sb.value == document.getElementById('<%=SearchTBHidden.ClientID%>').value) {
   document.getElementById('<%=searchTextBox.ClientID%>').value = "";
  }
 }

 function SearchTextNew(sb) {
  if (sb.value == "") {
   document.getElementById('<%=searchTextBox.ClientID%>').value =
    document.getElementById('<%=SearchTBHidden.ClientID%>').value;
  }
 }
//]]>
</script>
<div style="margin-bottom: 5px;">
 <table width="100%">
  <tr>
   <td class="ms-partline">
```

```
62          <div style="float: left; padding-top: 2px;">
63            <asp:updatepanel id="UpdatePanelNavigation" runat="server">
64            <ContentTemplate>
65            <asp:linkbutton runat="server" id="navigationMicroblog"
66              onclick="navigationMicroblog_Click"  style="margin-right: 4px;">
67              <asp:literal runat="server" Text="<%$Resources:microblog,
68                webpartnavigation_blog%>" />
69            </asp:linkbutton>
70            |
71            <asp:linkbutton runat="server" id="navigationAll"
72              onclick="navigationAll_Click"  style="margin-left: 4px; margin-right: 4px;">
73              <asp:literal runat="server" Text="<%$Resources:microblog,
74                webpartnavigation_all%>" />
75            </asp:linkbutton>
76            |
77            <asp:linkbutton runat="server" id="navigationUser"
78              onclick="navigationUser_Click" style="margin-left: 4px; margin-right: 4px;">
79              <asp:literal runat="server" Text="<%$Resources:microblog,
80                webpartnavigation_user%>" />
81            </asp:linkbutton>
82            </ContentTemplate>
83            <Triggers>
84              <asp:AsyncPostBackTrigger ControlID="navigationMicroblog" EventName="Click"
85                />
86              <asp:AsyncPostBackTrigger ControlID="navigationAll" EventName="Click" />
87              <asp:AsyncPostBackTrigger ControlID="navigationUser" EventName="Click" />
88            </Triggers>
89            </asp:updatepanel>
90          </div>
91          <div style="float: right;">
92            <asp:panel runat="server" ID="searchPanel" DefaultButton="searchButton">
93            <asp:textbox runat="server" borderstyle="Solid" borderwidth="1px"
94              bordercolor="Silver"
95            forecolor="Gray" text="<%$Resources:microblog, webpartsearch_text%>"
96              id="searchTextBox"></asp:textbox>
97            <asp:imagebutton runat="server" imagealign="Top"
98              imageurl="~/_layouts/images/Microblog/gosearch_20.png" id="searchButton"
99              onclick="searchButton_Click"></asp:imagebutton>
100           </asp:panel>
101         </div>
102       </td>
103     </tr>
104   </table>
105 </div>
106 <div style="clear: both;">
107   <asp:updatepanel id="UpdatePanelTweetsTable" runat="server"
108     UpdateMode="Conditional">
109     <ContentTemplate>
110       <center><asp:Label ID="noEntriesLabel" runat="server" Text="<%$
111         Resources:microblog, webpart_noentries %>" Visible="False"
112         ></asp:Label></center>
113       <asp:Repeater ID="repeaterTweetTable" runat="server"
114         onitemcommand="repeaterTweetTable_ItemCommand"
115         onItemDataBound="repeaterTweetTable_ItemDataBound">
116       <HeaderTemplate>
117         <table>
118       </HeaderTemplate>
119       <ItemTemplate>
120         <tr>
121         <td align="center" style="width:32px;">
122           <asp:Image ID="ProfileImage" runat="server" AlternateText='<%#
123             DataBinder.Eval(Container.DataItem, "AuthorPName") %>' ImageAlign="Middle"
124             ImageUrl='<%# DataBinder.Eval(Container.DataItem, "ProfileImageUrl") %>' />
125         </td>
126           <td>
```

```
127           <asp:HyperLink ID="profileLink" runat="server" Font-Bold="True"
128             NavigateUrl='<%# DataBinder.Eval(Container.DataItem, "ProfileUrl") %>'>
129           <%# DataBinder.Eval(Container.DataItem, "AuthorPName") + ":" %>
130           </asp:HyperLink>
131           <asp:Repeater ID="repeaterTweetMessage" runat="server"
132             onitemcommand="repeaterTweetMessage_ItemCommand">
133             <ItemTemplate>
134               <asp:HyperLink ID="userTag" runat="server" NavigateUrl='<%#
135                 DataBinder.Eval(Container.DataItem, "UserTagUrl") %>' Visible='<%#
136                 DataBinder.Eval(Container.DataItem, "UserTagVisible") %>'>
137                 <%# DataBinder.Eval(Container.DataItem, "UserTag") %>
138               </asp:HyperLink>
139               <asp:LinkButton ID="hashTag" runat="server" CommandArgument='<%#
140                 DataBinder.Eval(Container.DataItem, "HashTag") %>' CommandName="HashTag"
141                 Visible='<%# DataBinder.Eval(Container.DataItem, "HashTagVisible") %>'
142                 Text='<%# DataBinder.Eval(Container.DataItem, "HashTag")
143                 %>'></asp:LinkButton>
144               <asp:Literal ID="tweetText" runat="server" Text='<%# " " +
145                 DataBinder.Eval(Container.DataItem, "TweetText") %>'  Visible='<%#
146                 DataBinder.Eval(Container.DataItem, "TweetTextVisible")
147                 %>'></asp:Literal>
148             </ItemTemplate>
149           </asp:Repeater>
150           <asp:imagebutton ID="imageButtonDelete" runat="server"
151             AlternateText="<%$Resources:microblog, webparttweet_delete%>"
152             ImageUrl="~/_layouts/images/CNSREJ16.GIF" Width="12px" ForeColor="White"
153             ImageAlign="AbsBottom" Visible='<%# DataBinder.Eval(Container.DataItem,
154             "DeleteButtonVisible") %>' CommandArgument='<%#
155             DataBinder.Eval(Container.DataItem, "WebUrl") + "," +
156             DataBinder.Eval(Container.DataItem, "ID") %>'
157             CommandName="DeletePost"></asp:imagebutton>
158           <br />
159           <asp:label ID="tweetDateLabel" runat="server" text='<%#
160             DataBinder.Eval(Container.DataItem, "Created") + " " %>' Font-
161             Size="Smaller"></asp:label>
162           <asp:label ID="inSeperator" runat="server" text="<%$Resources:microblog,
163             webpartweblink_in%>" Font-Size="Smaller" Visible='<%#
164             DataBinder.Eval(Container.DataItem, "WebVisible") %>'></asp:label>
165           <asp:HyperLink ID="webLink" runat="server" Font-Bold="True"
166           NavigateUrl='<%# DataBinder.Eval(Container.DataItem, "WebUrl") %>'
167             Visible='<%# DataBinder.Eval(Container.DataItem, "WebVisible") %>' Font-
168             Size="Smaller">
169           <%# DataBinder.Eval(Container.DataItem, "Web") %>
170           </asp:HyperLink>
171         </td>
172       </tr>
173     </ItemTemplate>
174     <FooterTemplate>
175       </table>
176     </FooterTemplate>
177     </asp:Repeater>
178   </ContentTemplate>
179   <Triggers>
180     <asp:AsyncPostBackTrigger ControlID="moreTweets" EventName="Click" />
181     <asp:AsyncPostBackTrigger ControlID="tweetSubmit" EventName="Click" />
182     <asp:AsyncPostBackTrigger ControlID="navigationMicroblog" EventName="Click" />
183     <asp:AsyncPostBackTrigger ControlID="navigationAll" EventName="Click" />
184     <asp:AsyncPostBackTrigger ControlID="navigationUser" EventName="Click" />
185     <asp:AsyncPostBackTrigger ControlID="searchButton" EventName="Click" />
186     <asp:AsyncPostBackTrigger ControlID="repeaterTagCloudBlog"
187       EventName="ItemCommand" />
188     <asp:AsyncPostBackTrigger ControlID="repeaterTagCloudAll" EventName="ItemCommand"
189       />
190     <asp:AsyncPostBackTrigger ControlID="repeaterTagCloudUser"
191       EventName="ItemCommand" />
192   </Triggers>
```

```
193     </asp:updatepanel>
194     <asp:table runat="server" id="moreTweetsTable" style="margin-bottom: 5px"
195       width="100%">
196       <asp:TableRow runat="server">
197         <asp:TableCell runat="server" HorizontalAlign="Center">
198           <asp:UpdatePanel ID="UpdatePanelMoreTweets" runat="server">
199             <ContentTemplate>
200               <asp:linkbutton runat="server" id="moreTweets" onclick="moreTweets_Click">
201                 <asp:literal runat="server" Text="<%$Resources:microblog,
202                   webparttweets_more%>" />
203               </asp:linkbutton>
204             </ContentTemplate>
205             <Triggers>
206               <asp:AsyncPostBackTrigger ControlID="moreTweets" EventName="Click" />
207               <asp:AsyncPostBackTrigger ControlID="tweetSubmit" EventName="Click" />
208               <asp:AsyncPostBackTrigger ControlID="searchButton" EventName="Click" />
209             </Triggers>
210           </asp:UpdatePanel>
211         </asp:TableCell>
212       </asp:TableRow>
213     </asp:table>
214     <asp:table runat="server" width="100%">
215     <asp:TableRow runat="server">
216       <asp:TableCell runat="server" ColumnSpan="2">
217         <asp:UpdatePanel ID="UpdatePanelTweetInput" runat="server">
218           <ContentTemplate>
219             <asp:textbox runat="server" id="tweetInput" Width="100%" Rows="2"
220               TextMode="MultiLine"
221           onKeyUp="javascript:Count(this);" onChange="javascript:Count(this);"/>
222           </ContentTemplate>
223           <Triggers>
224             <asp:AsyncPostBackTrigger ControlID="tweetSubmit" EventName="Click" />
225           </Triggers>
226         </asp:UpdatePanel>
227       </asp:TableCell>
228     </asp:TableRow>
229     <asp:TableRow runat="server">
230         <asp:TableCell runat="server" HorizontalAlign="Left">
231           <asp:UpdatePanel ID="UpdatePanelTweetSubmit" runat="server">
232             <ContentTemplate>
233               <asp:button runat="server" text="<%$Resources:microblog,
234                 webpart_tweetSubmit%>" id="tweetSubmit" onclick="tweetSubmit_Click"
235                 enabled="False"/>
236             </ContentTemplate>
237             <Triggers>
238               <asp:AsyncPostBackTrigger ControlID="tweetSubmit" EventName="Click" />
239             </Triggers>
240             </asp:UpdatePanel>
241         </asp:TableCell>
242         <asp:TableCell runat="server" HorizontalAlign="Right">
243           <asp:UpdatePanel ID="UpdatePanelTweetInputLeft" runat="server">
244             <ContentTemplate>
245               <asp:Label runat="server" id="tweetInputLeft"></asp:Label>
246             </ContentTemplate>
247             <Triggers>
248               <asp:AsyncPostBackTrigger ControlID="tweetSubmit" EventName="Click" />
249             </Triggers>
250           </asp:UpdatePanel>
251         </asp:TableCell>
252     </asp:TableRow>
253     </asp:table>
254   </div>
255   <table width="100%">
256     <tr>
257       <td align="center" class="ms-partline">
258         <h3 class="ms-standardheader ms-WPTitle"
```

```
259       style="font-weight: normal; text-align: center;">
260         <asp:literal runat="server" Text="<%$ Resources:microblog,
261           webparttagcloud_header %>" />
262       </h3>
263     </td>
264   </tr>
265 </table>
266 <asp:updatepanel id="UpdatePanelTagCloud" runat="server" UpdateMode="Conditional">
267   <ContentTemplate>
268     <asp:Repeater ID="repeaterTagCloudBlog" runat="server"
269       onitemcommand="repeaterTagCloud_ItemCommand" Visible="False">
270       <HeaderTemplate>
271         <table width="90%" align="center">
272           <tr><td align="center" style="padding:5px">
273       </HeaderTemplate>
274       <ItemTemplate>
275         <asp:LinkButton ID="tagButton" runat="server" CommandArgument='<%#
276           DataBinder.Eval(Container.DataItem, "HashTagName") %>' CommandName="HashTag"
277           Text='<%# DataBinder.Eval(Container.DataItem, "HashTagName") %>' Font-
278           Size='<%# DataBinder.Eval(Container.DataItem, "FontSize") %>'
279           style="margin:5px"></asp:LinkButton>
280       </ItemTemplate>
281       <FooterTemplate>
282           </td></tr>
283         </table>
284       </FooterTemplate>
285     </asp:Repeater>
286     <asp:Repeater ID="repeaterTagCloudAll" runat="server"
287       onitemcommand="repeaterTagCloud_ItemCommand" Visible="False">
288       <HeaderTemplate>
289         <table width="90%" align="center">
290           <tr><td align="center" style="padding:5px">
291       </HeaderTemplate>
292       <ItemTemplate>
293         <asp:LinkButton ID="tagButton" runat="server" CommandArgument='<%#
294           DataBinder.Eval(Container.DataItem, "HashTagName") %>' CommandName="HashTag"
295           Text='<%# DataBinder.Eval(Container.DataItem, "HashTagName") %>' Font-
296           Size='<%# DataBinder.Eval(Container.DataItem, "FontSize") %>'
297           style="margin:5px"></asp:LinkButton>
298       </ItemTemplate>
299       <FooterTemplate>
300           </td></tr>
301         </table>
302       </FooterTemplate>
303     </asp:Repeater>
304     <asp:Repeater ID="repeaterTagCloudUser" runat="server"
305     onitemcommand="repeaterTagCloud_ItemCommand" Visible="False">
306       <HeaderTemplate>
307         <table width="90%" align="center">
308           <tr><td align="center" style="padding:5px">
309       </HeaderTemplate>
310       <ItemTemplate>
311         <asp:LinkButton ID="tagButton" runat="server" CommandArgument='<%#
312           DataBinder.Eval(Container.DataItem, "HashTagName") %>' CommandName="HashTag"
313           Text='<%# DataBinder.Eval(Container.DataItem, "HashTagName") %>' Font-
314           Size='<%# DataBinder.Eval(Container.DataItem, "FontSize") %>'
315           style="margin:5px"></asp:LinkButton>
316       </ItemTemplate>
317       <FooterTemplate>
318           </td></tr>
319         </table>
320       </FooterTemplate>
321     </asp:Repeater>
322   </ContentTemplate>
323   <Triggers>
324     <asp:AsyncPostBackTrigger ControlID="tweetSubmit" EventName="Click" />
```

```
325       <asp:AsyncPostBackTrigger ControlID="navigationMicroblog" EventName="Click" />
326       <asp:AsyncPostBackTrigger ControlID="navigationAll" EventName="Click" />
327       <asp:AsyncPostBackTrigger ControlID="navigationUser" EventName="Click" />
328     </Triggers>
329   </asp:updatepanel>
330   <asp:hiddenfield runat="server" value="<%$Resources:microblog, webpartsearch_text%>"
331     id="SearchTBHidden"></asp:hiddenfield>
332   <asp:updatepanel id="UpdatePanelNavigationTBHidden" runat="server">
333     <ContentTemplate>
334       <asp:hiddenfield runat="server" value="navigationMicroblog 1"
335         id="NavigationTBHidden"></asp:hiddenfield>
336     </ContentTemplate>
337   </asp:updatepanel>
```

Literaturverzeichnis

Alby, Tom: Web 2.0: Konzepte, Anwendungen, Technologien. – 3., übearb. Aufl. – München: Hanser, 2008

Alexa Internet, Inc.: Alexa Top 500 Global Sites. URL:<http://www.alexa.com/topsites>, 2010a, verfügbar am 20.05.2010

Alexa Internet, Inc.: Facebook.com Sit Info. URL: <http://www.alexa.com/siteinfo/facebook.com>, 2010b, verfügbar am 20.05.2010

Allen, Christopher <ChristopherA@LifeWithAlacrity.com>**:** Tracing the Evolution of Social Software. URL: <http://www.lifewithalacrity.com/2004/10/tracing_the_evo.html>, 2004, verfügbar am 15.03.2010

Alpar, Paul; Blaschke, Steffen (Hrsg.): Web 2.0 – Eine empirische Bestandsaufnahme. – 1. Aufl. – Wiesbaden: Vieweg+Teubner, 2008

Back, Andrea; Gronau, Norbert; Tochtermann, Klaus (Hrsg.): Web 2.0 in der Unternehmenspraxis: Grundlagen, Fallstudien und Trends zum Einsatz von Social Software. – 2., aktualisierte Aufl. – München: Oldenbourg, 2009

Baier, Toby; Weinreich, Harald; Wollenweber, Frank: Verbesserung von Social Navigation durch Identitätsmanagement. In: Mensch und Computer 2004: Allgegenwärtige Interaktion. – München: Oldenbourg, 9(2004), S.189-198

Bitkom <bitkom@bitkom.org>**:** Presseinformation. Nutzer im Schnitt länger als zwei Stunden pro Tag online. URL: <http://www.bitkom.org/files/documents/BITKOM-Presseinfo_Surfen_pro_Tag_01__05_2009.pdf>, 2009, verfügbar am 20.05.2010

Blood, Rebecca <rebecca@rebeccablood.net>**:** Weblogs: A history and perspective. URL: <http://www.rebeccablood.net/essays/weblog_history.html>, 2000, verfügbar am 11.03.2010

Boulton, Clint <clint.boulton@ziffdavisenterprise.com>**:** New Lotus GM Highlights Growth in Q1. URL: <http://www.eweek.com/c/a/Enterprise-Applications/New-Lotus-GM-Highlights-Growth-in-Q1/>, 2008, verfügbar 10.06.2010

Buhse, Willms; Stamer, Sören (Hrsg.): Enterprise 2.0: Die Kunst loszulassen. – Berlin: Rhombos, 2008

Coates, Tom <tom@plasticbag.org>**:** An addendum to a definition of Social Software. URL: <http://www.plasticbag.org/archives/2005/01/an_addendum_to_a_definition_of_social_software/>, 2005, verfügbar am 05.03.2010

Ebersbach, Anja; Glaser, Markus; Heigel, Richard; ..: Wiki: Kooperation im Web. – 2. Aufl. – Berlin: Springer, 2008

Elzer, Christoph <celzer@chip.de>**:** Kurz-URLs: Das riskante Spiel mit den Twitter-Links. URL: <http://www.chip.de/news/Kurz-URLs-Das-riskante-Spiel-mit-den-Twitter-Links_38459468.html>, 2009, verfügbar am 08.04.2010

Facebook, Inc.: Pressebereich. Statistiken. URL: <http://www.facebook.com/press/info.php?statistics>, 2010, verfügbar am 20.05.2010

Forrester Research, Inc.: The Forrester Wave™: Collaboration Platforms, Q3 2009. URL: <http://www.forrester.com/rb/Research/wave&trade%3B_collaboration_platforms,_q3_2009/q/id/47748/t/2>, 2009, verfügbar 10.06.2010

Friedman, Vitaly: Praxisbuch Web 2.0: Moderne Webseiten programmieren und gestalten. – 1. Aufl., 1. korrigierter Nachdruck – Bonn: Galileo Press, 2008

Garrett, James <jjg@ jjg.net>**:** Ajax: A New Approach to Web Applications. URL: <http://adaptivepath.com/ideas/essays/archives/000385.php>, 2005, verfügbar am 20.04.2010

Gross, Tom; Koch, Michael: Computer-Supported Cooperative Work. – München: Oldenbourg, 2007

Gutwin, Carl <gutwin@cpsc.ucalgary.ca>**; Greenberg, Saul** <saul@cpsc.ucalgary.ca>**; Roseman, Mark** <rosemang@cpsc.ucalgary.ca>**:** Workspace Awareness in Real-Time Distributed Groupware: Framework, Widgets, and Evaluation. URL: <http://www.markroseman.com/pubs/wahci96.pdf>, 1996, verfügbar am 15.03.2010

Heckner, Markus: Tagging, Rating, Posting: Studying Forms of User Contribution for Web-based Information Management and Information Retrieval. – Boizenburg: Werner Hülsbusch, 2009

IBM, Corporation: Lotus Connections: Social Software für Unternehmen. URL: <http://www-01.ibm.com/software/de/lotus/wdocs/connection/>, 2010, verfügbar am 10.06.2010

Kirchhof, Anja; Gurzki, Thorsten; Hinderer, Henning; ...: Was ist ein Portal?: Definition und Einsatz von Unternehmensportalen. – Frankfurt, Frauenhofer IAO – Competence Center Business Integration, Whitepaper, 2004

Klobas, Jane: Wikis: Tools for Information Work and Collaboration. – Oxford: Chandos Publishing, 2006

Koch, Michael; Richter, Alexander: Enterprise 2.0: Planung, Einführung und erfolgreicher Einsatz von Social Software im Unternehmen. – 2., aktualisierte und erw. Aufl. – München: Oldenbourg, 2009

Koch, Michael; Richter, Alexander: Funktionen von Social-Networking-Diensten. Proc. Multikonferenz Wirtschaftsinformatik, URL: <http://www.kooperationssysteme.de/docs/pubs/RichterKoch2008-mkwi-sns.pdf>, 2008, verfügbar am 22.03.2010

Liu, Lawrence: SharePoint: Yes, we've certainly come a long way! – URL: <http://blogs.msdn.com/b/sharepoint/archive/2007/12/27/sharepoint-yes-we-ve-certainly-come-a-long-way-and-happy-new-year-to-you.aspx>, 2007, verfügbar am 26.06.2010

McAfee, Andrew <amcafee@gmail.com>**:** Enterprise 2.0: The Dawn of Emergent Collaboration.URL: <http://sloanreview.mit.edu/the-magazine/articles/2006/spring/47306/enterprise-the-dawn-of-emergent-collaboration/>, 2006a, verfügbar am 22.03.2010

McAfee, Andrew <amcafee@gmail.com>**:** Enterprise 2.0, version 2.0.URL: <http://andrewmcafee.org/2006/05/enterprise_20_version_20/>, 2006b, verfügbar am 22.03.2010

Microsoft, Corporation: Microsoft Unveils SharePoint Server 2010 and Showcases New Functionality. URL: <http://www.microsoft.com/presspass/press/2009/oct09/10-19mssharepointconf09pr.mspx>, 2009, verfügbar am 26.05.2010

Microsoft, Corporation: Microsoft SharePoint 2010. URL: <http://sharepoint.microsoft.com/de-at/Seiten/default.aspx>, 2010a, verfügbar am 25.05.2010

Microsoft, Corporation: Microsoft SharePoint 2010. Produkt: Leistungsvermögen. URL: <http://sharepoint.microsoft.com/de-at/product/capabilities/Seiten/default.aspx>, 2010b, verfügbar am 15.05.2010

Microsoft, Corporation: ASP.NET-Webserver-Steuerelemente. URL: <http://msdn.microsoft.com/de-de/library/7698y1f0%28v=VS.80%29.aspx>, 2010c, verfügbar am 15.06.2010

Microsoft, Corporation: ASP.NET-Benutzersteuerelemente. URL: <http://msdn.microsoft.com/de-de/library/y6wb1a0e%28v=VS.80%29.aspx>, 2010d, verfügbar am 15.06.2010

Microsoft, Corporation: Übersicht über ASP.NET-Webseiten. URL: <http://msdn.microsoft.com/de-de/library/428509ah%28v=VS.80%29.aspx>, 2010e, verfügbar am 20.06.2010

Microsoft, Corporation: Übersicht über ASP.NET-Webparts. URL: <http://msdn.microsoft.com/de-de/library/hhy9ewf1%28VS.80%29.aspx>, 2010f, verfügbar am 20.06.2010

Microsoft, Corporation: Office Space: Features für SharePoint. URL: <http://msdn.microsoft.com/de-de/magazine/cc163428.aspx>, 2010g, verfügbar am 20.06.2010

Microsoft, Corporation: Developing SharePoint Applications: Glossary. URL: <http://msdn.microsoft.com/en-us/library/ee413946.aspx>, 2010h, verfügbar am 20.06.2010

Oleson, Joel <joleson@yahoo.com>**:** 7 Years of SharePoint: a History Lesson. URL: <http://blogs.msdn.com/b/joelo/archive/2007/12/28/7-years-of-sharepoint-a-history-lesson.aspx> 2007, verfügbar am 26.05.2010

O'Reilly, Tim <tim@oreilly.com>**:** What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. URL: <http://oreilly.com/web2/archive/what-is-web-20.html>, 2005, verfügbar am 05.03.2010

O'Reilly, Tim <tim@oreilly.com>; **Battelle, John:** Web Squared: Web 2.0 Five Years On. URL: <http://www.web2summit.com/web2009/public/schedule/detail/10194>, 2009, verfügbar am 25.06.2010

Pettey, Christy <christy.pettey@gartner.com>**:** Gartner Identifies the Top 10 Strategic Technologies for 2008. URL: <http://www.gartner.com/it/page.jsp?id=530109>, 2007, verfügbar am 22.03.2010

Raabe, Alexander: Social Software im Unternehmen: Wikis und Weblogs für Wissensmanagement und Kommunikation. – Saarbrücken: VDM Verlag, 2007

Richardson, Sharon <info@joiningdots.net>**:** SharePoint History. URL: <http://www.joiningdots.net/blog/2006/08/sharepoint-history.html>, 2006, verfügbar am 26.06.2010

Rubenstein, Benjamin <benjamin.rubenstein@neowin.net>**:** Microsoft launches Office and SharePoint 2010. URL <http://www.neowin.net/news/microsoft-launches-office-and-sharepoint-2010-retail-availability-in-june>, 2010, verfügbar am 20.05.2010

Schwichtenberg, Holger <buero@IT-Visions.abc>**:** Erläuterung des Begriffs Team Foundation Server 2010. URL: <http://www.it-visions.de/glossar/alle/3762/Team%20Foundation%20Server.aspx>, 2010, verfügbar 20.06.2010

Schönefeld, Frank: Praxisleitfaden Enterprise 2.0: Wettbewerbsfähig durch neue Formen der Zusammenarbeit, Kundenbindung und Innovation: Basiswissen zum erfolgreichen Einsatz von Web 2.0-Technologien. – München: Hanser, 2009

Teufel, Stefanie; Sauter, C; Muehlherr,T.; ..: Computerunterstützung für die Gruppenarbeit. – Bonn: Addison-Wesley.

Ullman, Chris <chris@cuasp.co.uk**>:** What is Ajax? URL: <http://www.wrox.com/WileyCDA/Section/id-303217.html>, 2007, verfügbar am 03.03.2010

Weil, Kevin <kweil@twitter.com>**:** Twitter Blog: Measuring Tweets. URL: <http://blog.twitter.com/2010/02/measuring-tweets.html>, 2010, verfügbar am 24.03.2010

Wikipedia: Wikipedia. URL: <http://en.wikipedia.org/wiki/Wikipedia>, 2010, verfügbar am 20.05.2010

Winer, Dave <scriptingnews1mail@gmail.com**>:** RSS 2.0 Specification. URL: <http://cyber.law.harvard.edu/rss/rss.html>, 2003, verfügbar am 04.03.2010

Winn, Phillip <pwinn@winn.com>**:** State of the Blogosphere: Introduction. URL: <http://technorati.com/blogging/article/state-of-the-blogosphere-introduction/>, 2009, verfügbar am 22.03.2010

Zahorsky, David <http://sharepoint.namics.com/2010/01/microsoft-share.html>**:** Namics SharePoint Weblog: Microsoft SharePoint 2010 Development. URL: <http://sharepoint.namics.com/2010/01/microsoft-share.html>, 2010, verfügbar 18.06.2010

Selbständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Thalheim, den 05.10.2010

Robert Müller